Enhancing Federated Learning Robustness and Fairness in Non-IID Scenarios

YANLI LI

Doctor of Philosophy

Supervisor: Dong Yuan
Associate Supervisor: Huaming Chen, Abubakar Sadiq Sani, Wei Bao

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Electrical and Computer Engineering
Faculty of Engineering
The University of Sydney
Australia

8 February 2024

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes. I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Student Name: Yanli LI Signature: Date: 2023/9/30

Abstract

Federated Learning (FL) is a distributed machine learning paradigm that allows multiple clients to collaboratively train a joint model without sharing the raw data. Despite its advantages, FL faces the security issues inherent to its decentralized nature, and FL clients often encounter unfair treatment from the design that prioritizes server interests. Furthermore, due to the clients' heterogeneity, the overall data distribution of FL is usually considered non-Independent and Identical (non-IID), which further brings challenges to FL robustness and fairness. Today, many studies have been proposed to mitigate the research gap; nevertheless, in the absence of a non-IID setting, ensuring robustness and fairness in FL remains an open problem. Therefore, in this thesis, we study several topics on the robustness and fairness of FL in non-IID scenarios, including attack surface reduction, poisoning attack defense, and implicit class-level fair enhancement.

We start by investigating FL's non-IID resource and propose the Mini FL framework. Based on a predefined grouping principle, Mini FL assigns similar clients to different groups and aggregates them respectively to achieve attack surface reduction. Then, we focus on defending against FL poisoning attacks, including Label Flipping Attacks and Class Imbalance Attacks. For the Label Flipping Attack, we introduce the HSCS FL method. It evaluates the accuracy of each class in both global and local models in each iteration. These accuracies are then translated into a score, and only clients with top scores are included in the current aggregation. For the Class Imbalance Attack, we introduce the Class-Balanced FL framework. This approach dynamically determines the aggregation weight for each client, considering their potential contribution to the current global model, thereby preventing the joint model biases toward specific data distributions. Lastly, we propose the ICB FL method to enhance FL fairness. This framework enables the server to identify implicit classes and dynamically distribute weights, ensuring a similar learning performance across these implicit classes. We provide mathematical proofs for each scheme and framework we proposed and conduct

experiments to evaluate their performance. The experimental results show that our methods effectively enhance the FL robustness and fairness in non-IID scenarios.

Authorship Attribution Statement

Chapter 3 of this thesis is accepted by the *Computers & Security*. This work is an extension of our previous Mini-FL scheme, which is published by the *Asian Conference on Computer Vision (ACCV 2022)*. I designed the study, mathematically proved the theories, and performed the simulations.

Chapter 4.1 of this thesis is under review by *Information Fusion*. I designed the study, mathematically proved the theories, and performed the simulations.

Chapter 4.2 of this thesis is under review by *Information Sciences*. I designed the study, mathematically proved the theories, and performed the simulations.

Chapter 5 of this thesis is accepted by the *Australian Symposium on Parallel and Distributed Computing (AusPDC 2023)*. I designed the study, mathematically proved the theories, and performed the simulations.

Student: Yanli LI Signature: Date: 2023/9/30

As the supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Supervisor: Dong Yuan Signature: Date: 2023/9/30

List of Publication

Accepted papers

Yanli Li, Abubakar Sadiq Sani, Dong Yuan, and Wei Bao. 'Enhancing federated learning robustness through clustering non-IID features.' In: *Asian Conference on Computer Vision (ACCV).* Springer. Dec 4-8 2022, Macau, China, pp. 41–55.

Yanli Li, Laicheng Zhong, Dong Yuan, Huaming Chen, and Wei Bao. 'ICB FL: Implicit class balancing towards fairness in federated learning.' In: *Australian Symposium on Parallel and Distributed Computing (AusPDC).* ACM. Jan 31–Feb 3 2023, Melbourne, Australia, pp. 135-142.

This paper has been awarded as the Best Paper in *AusPDC 2023.*

Yanli Li, Dong Yuan, Abubakar Sadiq Sani, and Wei Bao. 'Enhancing federated learning robustness in adversarial environment through clustering non-IID features.' In: *Computers & Security* 132.1 (2023), pp. 1–13.

Yanli Li, Chongbin Ye, Huaming Chen, Shiping Chen, Minhui Xue, and Jun Shen. 'Towards better ML-based software services: an investigation of source code engineering impact.' In: *IEEE International Conference on Software Service Engineering (SSE).* IEEE. Jul 2-8 2023, Chicago, USA, pp. 1-10.

Jehad Ibrahim, Yanli Li, Huaming Chen, and Dong Yuan. 'Holistic Evaluation Metrics for Federated Learning' In: *International Conference on Computer Supported Cooperative Work in Design (CSCWD)* IEEE. May 8-10 2024, Tianjin, China, pp. 1-6.

Papers that are under review

Yanli Li, Huaming Chen, Wei Bao, and Dong Yuan. 'Contribution-Wise byzantine-robust aggregation for class-balanced federated learning.' Under review at *Information Sciences*

Yanli Li, Huaming Chen, Wei Bao, and Dong Yuan. 'Honest score client selection scheme: Preventing federated learning label flipping attacks in non-IID scenarios.' Under review at *Information Fusion.*

Zhengmeng Xu, Hai Lin, Yufeng Chen and Yanli Li. 'Label noise robust crowd counting with loss filtering factor.' Under review at *Applied Artificial Intelligence.*

Zhengmeng Xu, Yujie Wang, Xiaotong Feng, Yilin Wang, Yanli Li, and Hai Lin. 'Quantum-Enhanced Forecasting: Leveraging Quantum Gramian Angular Field and CNNs for Stock Return Predictions' Under review at *International Journal of Forecasting.*

Acknowledgements

As I write this, my doctoral journey is drawing to a close. Throughout this academic path, I have received immense support from my supervisor team, family, and friends. I would like to sincerely express my gratitude to them here.

First and foremost, I want to thank my supervisor, Dr. Dong Yuan. During my studies, Dr. Yuan has provided me with selfless assistance and encouragement. While guiding me in academic exploration, Dr. Yuan also taught me how to face both academic achievements and setbacks. His rigorous academic approach, optimistic and open-minded attitude, and proactive research and life stance will forever influence and inspire me.

Secondly, I extend my sincere gratitude to my supervisor, Dr. Huaming Chen. Throughout my academic journey, whenever I faced challenges, Dr. Chen was always there to offer guidance with patience, engage in thorough discussions, and provide the crucial insights needed to propel my research forward. I am profoundly grateful for Dr. Chen's steadfast support over these years.

Furthermore, I want to thank Dr. Abubakar Sadiq Sani for guiding me in the fields of blockchain and security and Dr. Wei Bao for his guidance on distributed systems. Your profound insights and patient guidance have greatly enriched my research in these areas.

Lastly, I want to express my heartfelt thanks to my family and friends. Your warm support and encouragement have been the pillars upon which I leaned during challenging times. Without your constant presence and belief in my abilities, navigating the complexities of this academic endeavor would have been insurmountable.

This chapter of my life, though now drawing to a close, leaves behind a legacy of cherished moments, enduring relationships, and invaluable lessons learned. The warmth I've received, the bonds we've strengthened, and the memories we've created together are treasures that I will carry with me into the future. To every single person who has been a part of this remarkable journey, my gratitude is boundless.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Background and Motivation

1.1.1 Background of Federated Learning

Federated learning (FL) [65, 97] is a decentralized machine learning paradigm that allows multiple clients, such as smart devices, to collaboratively train a joint model under the guidance of a central server while ensuring that the training data remains decentralized. Owing the privacy-protecting nature, FL has attracted increasing attention from academia and industry, leading to its deployment across various domains [82, 108, 59, 120, 157]. In the typical FL [97, 85] iteration, the server first selects a group of participants and broadcasts the current global model. These participants then train the model locally using private data and subsequently transmit their local model updates (e.g., gradients) back to the server. Finally, the server averages the local gradients (called FedAvg [97]) and updates the global model to end the current learning iteration. Figure 1.1 illustrates the FL process.

Different from the centralized machine learning that assumes independent and identically (IID) training data [32], the overall data distribution of FL is usually considered as non-IID data distribution [65, 84]. This divergence is attributed to client heterogeneity, with each client corresponding to a specific user, a specific location, and a specific time window [89]. In the FL concept, non-IID data distribution can manifest in various forms [52, 84]: concept shift, wherein identical labels may possess distinct features; covariant or prior probability shift, carried by skewed feature or label distributions; and unbalancedness, wherein individual clients retain varying quantities of data.

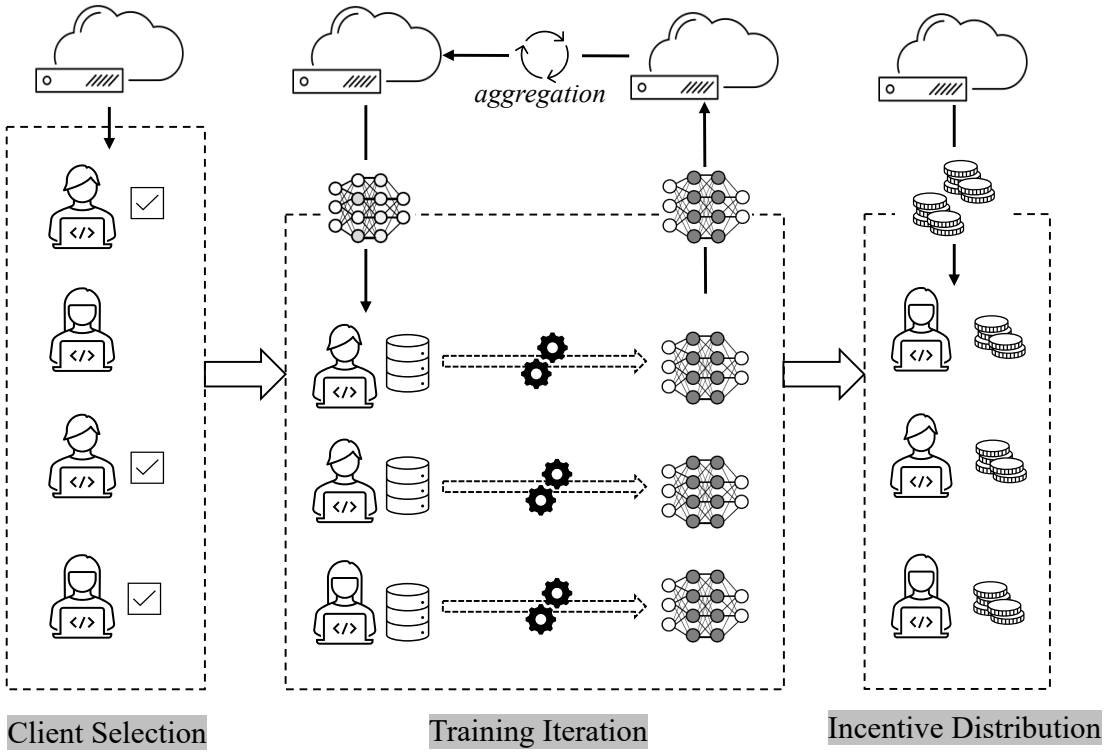

FIGURE 1.1: Federated learning process.

1.1.2 Motivation of Enhancing Federated Learning Robustness

Given that the FL task is solved by loose federation participating device [97, 65], a primary challenge of FL is clients may act maliciously[43, 103, 85], targeting the confidentiality, integrity, and availability of the global model. For instance, an adversary may leakage sensitive information to unauthorized users to break the FL system confidentiality [158], or craft the local training data to break the data integrity of FL [153, 136], or perform adversarial attacks when training the local model to break the availability of the FL global model [29, 99, 6]. Poisoning attacks, as one of the most representative adversarial attacks, have been widely studied by recent research [153, 37, 149, 179, 6]. By introducing perturbation and scaling the local model updates, the adversary can arbitrarily manipulate the global and degrade its performance, even breaking the global model entirely. Research [9] indicates that none of the linear combination rules are byzantine-resilient; a single malicious participant can potentially compromise the global model and consequently prevent it from convergence.

To defend against the various poisoning attacks, byzantine-resilient aggregation methods [9, 167, 45, 16, 110] have been introduced to replace the averaging aggregation in Vanilla FL. As introducing perturbation may lead the malicious gradient away and different from the benign, one avenue of defense research harnesses statistical features to detect manipulated model updates. These methods could select the median [167] of each model parameter or calculate the Euclidean (L2) distance [9] between each local model update to decide the most honest one as the aggregation outcome. On the other hand, the defender [16, 110] can evaluate the clients' model performance and consequently decide their aggregation weight to mitigate the impact of the suspect.

However, these FL methods generally assume the data distribution is IID [9, 167, 45]; when applied in non-IID settings, there can be a notable degradation in defense performance [89]. Compared to IID data distribution, non-IID introduces greater challenges in maintaining FL robustness. First, the non-IID data distribution introduces an expanded attack surface characterized by greater differences in clients' gradients. By colluding and staying close to the boundaries of benign clients, the adversary can keep stealth and consequently poison the global model. Second, because of the unbalancedness, some clients may naturally own less volume of data, resulting in a low-performance local model. While benign, these clients face challenges in participating in performance-based aggregation, and their information can not be learned by the global model effectively. Furthermore, given that FL clients might not possess the complete set of data labels, a fixed weight distribution mechanism could bias the global model's performance towards a specific data distribution.

1.1.3 Motivation of Enhancing Federated Learning Fairness

In FL, ensuring a fair learning environment presents another significant challenge [33, 169, 170, 87]. As most FL frameworks are designed in the interest of the FL server instead of clients, unfair treatment towards FL clients can emerge through the whole FL process [65]. For instance, clients achieving weak resources may have less chance to (i) be selected for the learning task, (ii) receive a model fitting their data distribution well, and/or (iii) receive reasonable incentives. Today, many fairness-aware works have been proposed to

enhance FL fairness from different perspectives, including client selection [57, 159], model optimization [87, 54, 28], contribution evaluation [66, 164] and incentive mechanism [172, 139, 175]. However, given the non-IID data distribution and its various expressions, the existing fair-aware model optimization methods still face limitations.

Owing to the non-IID nature of FL, clients may experience data unbalancedness, with varying data volumes across different classes, or encounter concept shifts where a single label can have multiple representations [65]. For example, in a handwritten digit recognition scenario, clients might possess varying quantities of numbers 1 to 9 (explicit class), and each digit could further encompass multiple representations (implicit class), like italic or floral font. The existing FL methods predominantly address the first type of non-IID situation, ensuring that each explicit class attains a similar learning performance. However, the effective identification of these implicit classes and the preservation of their fairness continue to pose challenges.

In non-IID scenarios, ensuring FL robustness and fairness remains an open problem. Therefore, we are driven to bridge this research gap by introducing several innovative solutions to bolster both robustness and fairness in non-IID FL scenarios. First, we investigate the main non-IID resource of FL. Based on the insight, we devise a clustering-based aggregation method that groups similar clients, aiming to reduce the overall attack surface. Second, we explore the scenarios when clients owing imbalanced training data encounter various poisoning attacks. We introduce several new robust aggregation mechanisms that monitor each class learning performance of the current model and dynamically distribute the aggregation weight to prevent Label Flipping Attacks and avoid the global model bias to a particular data distribution. Last, we study the fairness-enhancement mechanism when implicate classes exist in the learning environment. We propose a novel FL framework that enables the server to identify the implicit classes and achieve implicit class-level fairness.

1.2 Contribution

Each of our proposed frameworks aimed at enhancing the robustness and fairness of federated learning in non-IID scenarios makes practical contributions. Here is a brief summary of their individual contributions.

1.2.1 Mini Federated Learning

Mini Federated Learning (FL) framework is designed to reduce the large FL attack surface carried by its non-IID nature. Compared with the existing byzantine-resilient FL methods, Mini FL introduces a grouping principle based on a selected grouping feature and consequently performs group-based aggregation. In this context, we identify Time-feature, Geo-feature, and User-feature as the primary sources of non-IID in FL and utilize them as potential grouping feature candidates. We provide a formal derivation to demonstrate the reduction in attack surface when Mini FL defends against a state-of-the-art attack, compared to the existing FL method. We comprehensively evaluate our Mini FL when defending against various attacks in different non-IID degrees; the experimental results show our Mini FL achieves a higher testing accuracy compared to the existing FL methods.

To further demonstrate the practicality of the proposed Mini FL, we provide a case study with a real-world dataset. The results show Mini FL can achieve a lower loss value compared with the benchmark, which effectively enhances the robustness of the FL.

Our MiniFL is an extension of our previous work, *Enhancing Federated Learning Robustness Through Clustering non-IID Features*, which was published at the *2022 Asian Conference on Computer Vision (ACCV)*. Our Mini FL framework is published at *Computers & Security* under the name *Enhancing Federated Learning Robustness in Adversarial Environment through Clustering Non-IID Features*.

1.2.2 Honest Score Client Selection based Federated Learning

Honest Score Client Selection-based Federated Learning (HSCSFL) is designed to defend against the FL Label Flipping Attack in non-IID scenarios. It achieves robustness by implementing an innovative client selection scheme called the Honest Score Client Selection Scheme (HSCS). HSCS relies on a clean evaluation dataset to monitor the performance of the current global model and identify the class(es) that are most susceptible to being attacked in each iteration. Clients with low testing accuracy on susceptible classes are deemed less trustworthy and receive a lower honesty score; only the clients achieving top honest scores are included in the subsequent aggregation process. We conduct a comprehensive evaluation of our proposed HSCSFL using diverse datasets with varying degrees of non-IID characteristics and employing different Label Flipping Attacking strategies; our experimental results demonstrate the HSCSFL algorithm effectively enhances the robustness of FL and can defend against Label Flipping Attacks in non-IID scenarios.

Our HSCSFL algorithm is currently under review at *Information Fusion* under the name *Honest Score Client Selection Scheme: Preventing Federated Learning Label Flipping Attacks in Non-IID Scenarios*.

1.2.3 Class-Balanced Federated Learning

Drawing inspiration from the limitation of the existing local model performance-based robust aggregation methods, we introduce a pioneering attack strategy called the Class Imbalance Attack. In contrast to conventional poisoning attacks that introduce perturbations, our Class Imbalance Attack generates malicious gradients crafting training data distribution. The crafted training data covers most classes to cheat a heavyweight in the aggregation but selectively excludes a class to prevent the global model from learning any information pertaining to that particular class.

To defend against the Class Imbalance Attack and enhance the robustness of FL, we introduce the Class-Balanced FL method based on a novel contribution-wise byzantine-robust aggregation rule. In the proposed algorithm, the server itself keeps a small dataset and maintains

a model known as the server model. During each training iteration, the server dynamically assigns an honest score and a contribution score to each client based on the server model's evaluation. These scores are consequently combined to decide the clients' weight in aggregation and generate the robust model update. We conducted various experiments on different datasets to evaluate the robustness of our approach, and the empirical results demonstrate the effectiveness of our proposed Class-Balanced FL method.

Our Class-Balanced FL algorithm is currently under review at *Information Sciences* under the name *Contribution-Wise Byzantine-Robust Aggregation for Class-Balanced Federated Learning*.

1.2.4 Implicit Class Balanced Federated Learning

The presence of data heterogeneity can lead to different representations for a single label (explicit class), often referred to as different implicit classes. Implicit Class Balanced Federated Learning (ICB FL) is designed to enhance implicit class-level fairness in federated learning. The ICBFL employs a novel training scheme, namely the Single Class Training Scheme (SCTS), which asks each client to use one class data training model in each iteration. Through performing SCTS, the server can then perform unsupervised learning to effectively recognize implicit classes, assign balanced weights to each client, and accomplish implicit class-level balancing. We evaluate our ICBFL through different datasets; the experimental results show our proposed ICB FL can effectively identify the implicit class and enhance both explicit and implicit class level fairness.

Our ICB FL framework is published at *2023 Australasian Symposium on Parallel and Distributed Computing* under the name *ICB FL: Implicit Class Balancing Towards Fairness in Federated Learning*.

1.3 Organization

In this thesis, we study the robustness and fairness topics of federated learning in non-IID scenarios and propose several optimized FL algorithms. The sections of the thesis are structured as follows, with a visual representation provided in Figure 1.2.

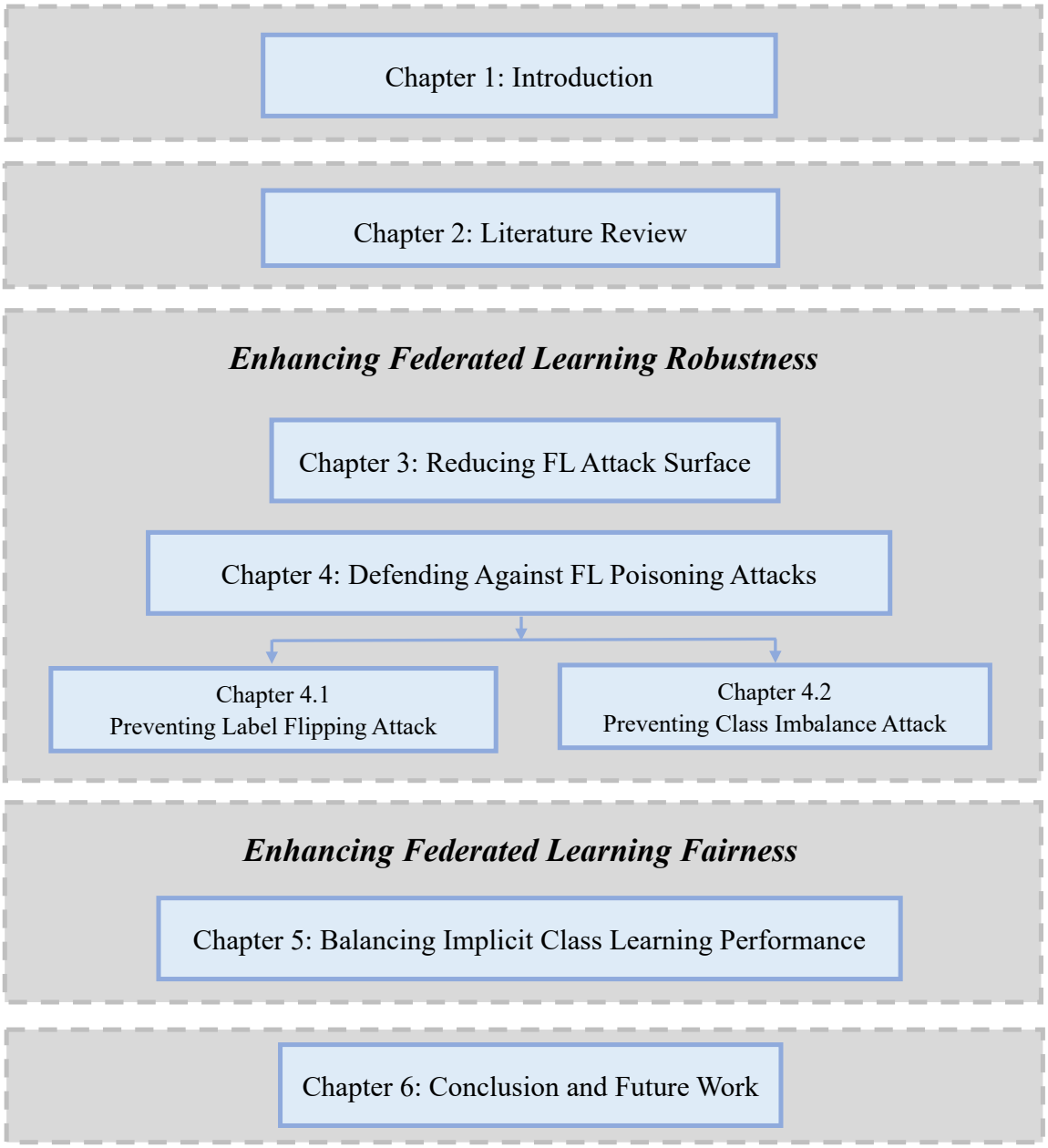

FIGURE 1.2: Thesis outline.

In Chapter 2, we summarise the literature review of the related topics, including FL adversarial attacks, defense framework, and fairness-enhanced FL methods. Chapter 3 and Chapter 4 focus on the FL robustness enhancement under non-IID settings. In Chapter 3, we propose "Mini FL," a solution for achieving a smaller attack surface in FL through clustering non-IID features. In Chapter 4, we propose solutions for defending against FL poisoning attacks. Specifically, we propose "HSCSFL," a client selection scheme based FL to prevent Label Flipping Attacks in Chapter 4.1, and "Class Balanced FL," a solution for guaranteeing the global model achieves balanced learning performance across different and defends against Class Imbalance Attacks in Chapter 4.2. Next, Chapter 5 proposes "ICB FL," a novel algorithm for identifying implicit class and granting the implicit class level fairness for FL. Finally, we wrap up the thesis with the conclusion and future work in Chapter 6.

CHAPTER 2

Literature review

2.1 Overview

In this literature review, we survey the current state-of-the-art byzantine attacks of federated learning (FL) and frameworks that enhance FL robustness and fairness. Firstly, we survey the existing adversarial attacks in FL to illustrate the threat model for Chapter 3 and Chapter 4. Secondly, to defend against these attacks, the byzantine-resilient FL method is required. We survey the state-of-the-art FL defense frameworks, which provide the base for Chapter 3 and Chapter 4. Finally, we investigate the existing fair-enhanced FL studies to provide the background and information for Chapter 5.

2.2 Adversarial Attacks in FL

Compared to data-centralized collection learning schemes, FL provides a different way of training models collaboratively [97, 65]. Instead of relying on a visible and centralized collection dataset, federated learning trains the model across a potentially vast array of unreliable devices, each equipped with private, uninspectable datasets. Although FL potentially preserves the privacy of participants, the local training process introduces a new attack surface and brings opportunities for potential attacks [179, 37, 149, 153]. Specifically, as data remains on the devices and only model updates are shared, adversaries could attempt to exploit this setup to compromise the model's integrity, privacy, or performance.

2.2.1 Poisoning Attacks

As one of the most representative adversarial attacks, poisoning attacks attempt to degrade the model performance or totally destroy the global model. Depending on the stage at which the poisoning occurs, poisoning attacks can be further categorized into model poisoning attacks and data poisoning attacks.

A) Model poisoning Attacks

Model poisoning attacks refer to adversaries directly manipulating reports (local model updates) submitted to the service provider. One of the most straightforward strategies for executing model attacks involves introducing a fixed perturbation or substituting the benign gradient parameters with fixed values. For instance, Reverse Attack [29], and Random Attack [99] generate the malicious gradient by reversing and randomly replacing the original gradients. Under the Partial Drop Attack [99], the adversary replaces a preset percentage of the benign gradient parameters as 0; the variants of Partial Drop Attack enlarge the attack effectiveness by using other malicious values if the benign gradients naturally carry a large amount of 0. Although these attack strategies can effectively degrade the global model performance and decrease the testing accuracy, the malicious gradients are unavoidably away from the benign gradients, which can be identified and discarded by the byzantine resilient FL methods.

To enhance the attacking performance and avoid being identified by the defender, some existing attacks [179, 149, 37] design more dynamic attack strategies. For instance, Little is Enough Attack [6] first normally trains the model through the data of the clients controlled. Then, the adversary statistics the mean value and standard deviation (Std) of the gradients owned. Based on the statistics, the adversary subsequently generates the crafted gradient by adding a scaled Std on the mean value. The Fall of Empires Attack [154] follows a similar strategy, which generates the malicious gradient by timing the mean value with a preset scalier. To maintain the stealth of model poisoning attack, research [179] proposes an optimization-based model poisoning attack, which only manipulates a small fraction of the local model updates instead of all gradient parameters. Under the study [179], adversary

training is organized into two tasks: the main task and the adversarial task. In the main task, the adversary trains the model in the usual manner. Conversely, in the adversarial task, the adversary embeds adversarial features into the neural network's redundant space to enhance the attack's persistence.

Recently, Local Model Poisoning Attack [37] proposes a more flexible attacking algorithm that adopts the attack strategy for different defense frameworks. This attack first infers the optimal global model updates and subsequently formulates the optimization problem that aims to reverse the global model update the most along the infers. This work [37] applies the attack in Krum [9], Trimmed-mean [167], and Median [167] frameworks and consequently generates the Krum Attack and Trim Attack. The study [149] targets the Krum framework with an emphasis on evasion detection, specifically through covert model poisoning (CMP). Research [149] first formulates the model poisoning as an optimization problem, aiming to minimize the Euclidean distance between the manipulated model and a designated one, all within the constraints set by Krum. Based on the solution to the optimization problem, [149] subsequently formulates CMP algorithms to counter the Krum framework. To improve the practicability, a low-complexity CMP algorithm is introduced and achieves the optimization complexity reduction.

B) Data poisoning Attacks

Different from model poisoning attacks that directly introduce the perturbation to the gradient, data poisoning attacks generate crafted gradients by training models based on poisoned data. Label Flipping Attack is one of the most representative data poisoning attacks. In Label Flipping Attacks, the adversary mismatches the training labels and training data to generate the malicious gradient, aiming to degrade the attacked class learning performance of the joint model. This study [153] considers the support vector machine (SVM) algorithm and aims to find a label-flipping combination under a given budget so that a classifier trained on such data will have maximal classification error. [153] proposes several flipping strategies to attack SVM, including Uniform random flip, Nearest-first flip, Furthest-first flip, and Adversarial Label Flips Attack on SVMs (ALFA), which shows the effectiveness by dramatically changing the decision boundaries. Research [136] introduces the Label Flipping Attack in the deep

learning scenarios and evaluates the key conditions for successfully performing attacks, including sufficient adversarial participants and continuous participation. To enhance the flexibility of the attacks, study [78] proposes the On-Off Label Flipping strategy. Under the On-Off Label Flipping strategy, the adversary first acts as a benign client for a period of time and builds a positive expectation in the defense system to strengthen the impact of the following updates. Then, the adversary betrays the system and performs an attack for a subsequent period of time. As the betrayal behavior lowers the reputation, the adversary may toggle back to acting as the benign client again after the One-Off attack and recover its reputation.

Label Flipping Attack also shows its significant effectiveness in the real-world applications. For instance, Study [134] implements this attack within a malware detection task for Android platforms, employing a Silhouette Clustering-based Label Flipping Attack (SCLFA). This method involves attackers calculating the silhouette scores of data samples and targeting those with negative values for generating polluted data through label flipping. Similarly,[174] illustrates the potency of Label Flipping Attacks in compromising spam filtering systems. By executing entropy-based flipping attacks, their study managed to elevate the false negative rate of Naive Bayes classifiers in the presence of label noise, without impairing the classification of legitimate emails. Furthermore, [125] explores the impact of Label Flipping Attacks on hardware Trojan detection systems, demonstrating how model performance could be substantially degraded through a stochastic hill-climbing search-based flipping strategy, incurring minimal costs for label manipulation. These instances underscore the critical need for robust defenses against Label Flipping Attacks in diverse application areas.

C) Backdoor Attacks

Backdoor Attack is a special type of poisoning attack, which aims to lead the global model to misbehave only on a selected minority of examples while maintaining good overall accuracy on all other examples [41, 90]. Recent studies indicate that the Backdoor Attack can be executed either by poisoning the clean training data or by introducing crafty perturbations to benign gradients. In other words, the Backdoor Attack can be categorized as either a model poisoning attack or a data poisoning attack based on the poisoning strategy.

When performing Backdoor Attacks through the data poisoning strategy, the adversary should first decide on a trigger (backdoor) and consequently assign a selected label for the trigger carrier to form the poisoning data. This backdoor could be semantic or artificial. For semantic backdoor, study [5] leverages the samples with uncommon attributes as the backdoor, such as the cars with unconventional colors (e.g., green), scenes containing a unique object (e.g., a striped pattern), or trigger sentences in word prediction problems that conclude with an attacker-selected target word. Similarly, research [107] proposes an attack strategy to introduce a backdoor in the FL-based IoT intrusion detection system. In this scenario, the adversary targets packet sequences originating from specific malware-driven malicious traffic. Different from the semantic backdoor that replies on existing share properties, the artificial backdoor manually poisons benign samples by introducing triggers. Study [85] demonstrates the adversary can manually add pattern "L" at the images' corner to activate the backdoor, where the pattern "L" does not naturally exist in the benign samples. A recent study [4] extends the scope of backdoor attacks to real-world settings. The research reveals that beyond digital triggers, physical triggers such as sunglasses, tattoos, and earrings can also serve as triggers for organizing backdoor attacks.

Within the Vanilla FL framework, the most straightforward method to execute a backdoor attack using a model poisoning strategy is scaling the malicious updates containing backdoor information to dominate updates from benign clients. [4] first employs the replace method, in which the attacker seeks to substitute the new global model with a poisoned one by a wisely chosen factor. While the scaling-replacement method proves effective in averaging aggregation, straightforward scaling appears to be naive to success under clipping and restricting defenses. To enhance the stealth of model poisoning attacks, [146] proposes a projected gradient descent (PGD) attack strategy, which projects the crafted model on a small ball centered around the global model from the previous iteration.

2.2.2 Free-Riding Attacks

In FL, malicious clients can gain advantages (e.g., the joint model, incentive, etc.) from the training tasks without contributing their local computing or data resource [143, 39]. Although

these free-riding attackers do not deliberately poison the model, the free-riding may result in insufficient computing resources and training data and subsequently lead to degrading learning performance.

Research [39] introduces a Free-riding Attack framework based on the Vanilla FL and shows that the adversary can perform a Free-riding Attack in each learning iteration. To enhance its stealthiness, [39] adds noise to the constructed parameter updates and applies Stochastic Gradient Descent (SGD) to maximize similarity with updates from other benign clients. Research [143] proposes a novel Free-riding Attack in which the model is trained locally using a small dataset masqueraded as a large dataset to obtain more incentive rewards. Study [111] introduces a Free-riding attack strategy, namely Random Weights Attack. Under this attack, the adversary creates a gradient update matrix with dimensions matching the received global model by randomly sampling each parameter from a uniform distribution within a pre-designed range.

2.2.3 Sybil Attacks

The Sybil Attack refers to a single attacker (or a small amount attackers) joining the system with multiple colluding identities to enhance the stealth and effectiveness of an adversarial attack. Given that traditional FL permits participants to freely join and exit, it becomes inherently susceptible to Sybil Attacks.

The research [140] first introduces Sybil Attacks within the FL context. It demonstrates Sybil Attacks can significantly enhance the impact of other attacks (such as the Label Flipping Attack) by misleading the global model into classifying handwritten digits “1” as “7” through only two sybils. Study [18] builds upon the findings of [140], delving deeper into the vulnerability of FL to Sybil Attacks, as well as exploring associated attack strategies and objectives. This work [18] categorizes and names the Denial-of-Service (DoS) attacks instigated by Sybil Attacks as Training Inflation and demonstrates Sybil Attacks can be performed from various FL dimensions. From the data distribution perspective, Sybils can use identical datasets, different datasets, or synthetic data to train local models and generate

updates. On the other hand, to dominate the benign clients and amplify their influence on the system, Sybils might coordinate amongst themselves before producing the subsequent series of model updates, achieving different levels of coordination.

2.2.4 Communication Layer Attacks

FL continuously updates the joint model, which relies on communications between the server and multiple participants. However, the large volume of data exchange and frequent communication drives a significant communication bottleneck and brings new attack interfaces for adversaries.

The man-in-the-middle (MiTM) attacks [2], as one of the most representative network layer attacks, has been introduced in FL by recent studies [165, 142]. As FL leverages the single server to aggregate model updates and organize the learning process, performing MiTM between the server and participants can effectively block the server and lead to a single point of failure. Research [162] explores attacking the FL communication bottleneck by decreasing the bandwidth, delaying the response, and increasing the struggle possibility. Through performing attacks on the communication of FL, [162] shows the model convergence and performance can be degraded significantly.

2.2.5 Evasion Attacks

In evasion attacks, the adversary deliberately adds slight malicious perturbations to input samples during the inference phase, causing the classifier to misclassify the sample with a high probability. One of the most representative works of the evasion attack has been introduced by [42], which is a so-called adversarial example. By adding unnoticeable pixel perturbation on a panda image, [42] successfully changes the classification of GoogLeNet [133] from panda to gibbon with 99.3% confidence.

Based on the information owned by the adversary, the evasion attacks could be further divided into white-box and black-box attacks. In the white-box setting, the adversary has full

knowledge of the learning algorithm and model parameters and can craft adversarial examples based on the information. For instance, studies [96, 76] generate adversarial examples by maximizing the loss function, subject to a norm constraint, using constrained optimization techniques like projected gradient ascent. On the other hand, the black-box setting refers to the attackers having no knowledge about the algorithm and parameter information of the FL system and generating adversarial samples through the interaction process or query access with the system. Considering the black-box setting, [22] proposes a zeroth order optimization (ZOO) based attack to estimate the target model updates and generate adversarial examples consequently. Similarly, Boundary Attack [13] has been designed in the same scenario. When performing Boundary Attack [13], the adversary first identifies the boundary separating adversarial and non-adversarial samples, then subtly introduces perturbations along this boundary to produce adversarial examples.

A recent work [109] introduces an Evasion Attack in the Vertical FL (VFL) scenarios and proposes the Adversarial Dominating Inputs Attack. Unlike the aforementioned adversarial samples that domain the whole feature space, the Adversarial Dominating Inputs Attack [109] can dominate the inputs of other clients by controlling a fraction of the feature inputs and lead to the misclassification of specific inputs. In the meanwhile, the Adversarial Dominating Inputs Attack can also result in reduced and unfair incentive rewards for impacted participants due to their decreasing contributions.

2.3 Defence Frameworks in FL

As the diversity and complexity of adversarial attacks against FL increase, new defenses are being developed to counteract their malicious impacts. These defense frameworks generally follow the setting that the server is honest and can play the defender role. As some defense methods are effective against multiple types of attack categories, we survey these FL defense frameworks from different technologies instead of grouping them according to the attack category defended.

2.3.1 Data Sanitization

Data sanitization has been introduced to remove malicious and suspicious data before the training process to defend against the Data Poisoning Attacks in the FL settings. The research [27] introduces a novel training pipeline that incorporates a data sanitization phase. During each iteration, this phase creates multiple models, termed micro-models, based on subsets of the training data. These micro-models are used to produce provisional labels for each training input and consequently combined in a voting scheme to determine which parts of the training data may be potentially attacked. From the data shuffling perspective, study [30] proposes a byzantine-resilient matrix-vector (MV) multiplication and further proposes an algorithm based on the data encoding process and error correction over real numbers to combat various adversarial attacks.

Although data sanitization can filter out some potential attacked data and keep training data clean, it relies on directly accessing the local data of clients, which may raise new privacy issues. A recent work [74] indicates that data sanitization can only work in cross-silo FL with strong authentication and be performed locally by the client itself.

2.3.2 Anomaly Detection

Anomaly detection refers to the defender analyzing the client's learning pattern through statistical technology [163, 98, 51]. Once unexpected patterns, abnormal behaviors, or anomalous data are detected, the defender will issue a warning and take responsive measures. The FL anomaly detection is generally performed from client and data two perspectives.

A) Client Anomaly Detection

By employing a pre-trained auto-encoder model, study [163] introduces Auto-encoder-Based Anomaly Detection to detect abnormal model weight updates from the clients and consequently erase the negative impact carried by the potential attackers. Inspired by spectral anomaly detection that distinguishes abnormal data instances by capturing the normal data features, research [135] designs a novel robust FL framework. The spectral anomaly detection model is trained through a centralized training process and applied in the following learning

iteration. Since the detection threshold is adjusted dynamically in each communication round, it becomes challenging for the adversary to understand the framework, ensuring they can be effectively excluded from the aggregation. Recent research [98] presents a visualization scheme for anomaly detection, introducing VADAF. Specifically, VADAF captures the intricate dynamics of the FL training process, aiding in investigating various client issues and assessing the severity of anomalous clients.

B) Data Anomaly Detection

Anomalous data, potentially introduced during the data sampling process, can adversely affect the model's training performance. Consequently, data anomaly detection has been introduced to identify outliers in the dataset or values that are far from the normal data feature values. Research [23] proposes a two-phase iterative adversarial detection method to identify software samples in the malicious application detection system that have been subjected to poisoning attacks. Research [69] proposes an anomaly detection method for time series datasets based on recursive auto-encoding, which reduces the impact of overfitting to anomalies. To further improve robust and efficient anomaly detection in unsupervised time series scenarios, a variational recurrent encoder model [68] can separate anomalies from normal data without relying on anomaly labels. However, data anomaly detection suffers from difficulty similar to data sanitization; the centralized data detection process may introduce significant privacy risks and communication costs.

2.3.3 Adversarial Training

During the model training process, a minor perturbation can be manually introduced to enhance the model's robustness, namely Adversarial Training. Research [138] introduces ensemble adversarial training in centralized learning settings, which augments training data through transferring perturbations from other pre-trained models.

Recent studies [81, 51, 20] extend the adversarial training in decentralized settings. Specifically, study [81] applies adversarial training in the FL environment to reduce model drift and accelerate model convergence. Nevertheless, as the perturbation introduction inevitably

affects the classification accuracy, the performance carried by adversarial training may not be stable against complex black-box attacks. On the other hand, adversarial training relies on a massive volume of training datasets, which increases the cost of computation resources on the local devices. Especially in cross-device FL environments with multi-participants, lightweight clients can not afford the high costs of adversarial training. Research [51] proposes a novel learning scheme that propagates adversarial robustness from high-resource users, who can undertake adversarial learning, to low-resource users throughout the FL process. Research [20] investigates the effectiveness of defencing Evasion Attacks through adversarial learning. Specifically, Gaussian noise is used to smooth the training data by including adversarial data in the training dataset.

2.3.4 Model Compression

The large model size enables the DL model to achieve a good learning performance but also introduces both communication challenges and a broadened attack surface when deploying the model in the FL environment. To reduce communication overhead and parameter redundancy, model compression technology has been widely investigated by recent studies [112, 121, 92].

As one of the model compression technologies, knowledge distillation has been incorporated into transfer FL in research [121], which introduces a general framework named FedMD. FedMD allows clients with different computational resources to design different network structures, protecting data privacy and enhancing the performance of local models. While FedMD improves the local model performance, the participants are required to devote a portion of data privacy to form a shared dataset. Following FedMD [121], research [92] leverages unlabeled data from local model outputs to achieve model fusion, which further accelerates model convergence and mitigates data privacy leakage.

Pruning is another model compression technique that can eliminate redundant or poisoned model neurons. Research [93] introduces a new pruning technique through trimming and fine-tuning the model returned by attackers to defend against backdoor poisoning attacks on the training set and communication bottleneck attacks. Recent research [64] proposes a PruneFL

method with adaptive and distributed parameter pruning in FL settings. Through employing PruneFL, the model can achieve a similar learning performance while reducing computational costs and shortening the learning time. Inspired by the coding theory [141], [21] proposes the DRACO framework for robust distributed training. In the DRACO [21] framework, each client evaluates redundant gradients, which the parameter server subsequently employs to counteract the impact of adversarial updates. Similarly, DETOX [117] has also been introduced based on redundancy to provide a more robust gradient aggregation. DETOX algorithm [117] includes two steps: a filtering step to reduce the effect of byzantine nodes by using limited redundancy and a hierarchical aggregation step that could be used in tandem with other existing robust aggregation methods.

2.3.5 Emerging Defense Frameworks

Although most byzantine-resilient FL methods [65] follow the setting that the server (service provider) acts as the defender role, some works consider defending the poisoning attacks from the client's perspective. A recent work [131] proposes a novel algorithm to enhance FL robustness from the participants' side, namely White Blood Cell for Federated Learning (FL-WBS). This study [131] first introduces the Attack Effect on Parameter (AEP) to evaluate the model poisoning attacks' impact on global model parameters and subsequently introduces the FL-WBS framework to prevent AEP from being hidden in the kernels of Hessian matrices on benign devices.

Owing the anonymity and distributed characteristics, enhancing FL robustness through blockchain has become an appealing research trend [105, 115]. Research [70] considers mitigating the one-point failure (server be controlled by malicious) of FL by introducing blockchain technology in the current FL algorithm. Specifically, research [70] proposes BlockFL, which replaces the server with a blockchain to relieve the server security issue and achieve decentralization. In each iteration, the clients normally train the model and then broadcast the gradients to the miners. The miners cross-verify the gradients and generate the candidate block until a preset block size is achieved. Once the block is published, the clients download the block and locally compute the global model update and consequently utilize

the outcome to update the model. Study [38] proposes a reputation-based FLchain system, namely BAFL, to achieve asynchronous FL strategy and maintain FL robustness. Under BAFL [38], the training time, training data size, gradients correlation, and global update cheating times are generated by the entropy weight method and are recorded in the blockchain to achieve long-term trust.

Some works [91, 178, 119] consider introducing the defense framework from traditional ML scenarios to FL settings. For instance, PDGAN [178] generates testing datasets through GAN technology to defend against Data Poisoning Attacks. Research [182] proposes Moving Target Defense to increase attacking cost and complexity by changing a system to reduce or move the attack surface available for exploitation by the adversary. However, most of these studies [182, 119] may show effectiveness in Horizontal FL (HFL) settings but face new challenges when deploying in Vertical FL (VFL) scenarios.

Besides the aforementioned methods, a more effective strategy introduces the defense in the aggregation step, namely byzantine-resilient aggregation. Instead of relying on averaging aggregation, these FL methods limit the impact of the adversary by assigning a reduced weight to the suspicious. In this thesis, we mainly investigate and propose byzantine-resilient aggregation methods for enhancing FL robustness; we survey the existing byzantine-resilient aggregation methods in the next subsection.

2.3.6 Byzantine-Resilient Aggregation

Linear combination rules, including averaging, have been demonstrated to lack resilience against byzantine attacks [9]. In particular, a single malicious worker can corrupt the global model and even prevent global model convergence. To defend against various attacks, byzantine-resilient aggregation methods have been proposed to replace the averaging aggregation and enhance FL robustness.

A) Statistical Feature and Similarity-Based Robust Aggregation

When introducing perturbation in the model updates, malicious updates are unavoidable to deviate more from other normal updates in FL. Based on this insight, some research designs statistical-based aggregators on the server side to aggregate the most honest clients' gradients and keep the global model away from potential malicious updates. For instance, Krum (multi-Krum) [9] regards the client (could be a small number of clients) that is closest to all participants as the most honest and uses its (them) gradient as the aggregation outcome. Specifically, Krum first calculates the L2 (Euclidean) distance between the gradient to others and consequently ranks the distance. The gradient achieving the shortest distance value will act as the global gradient and update the global model. Trimmed mean [167] and Median [167] enhance the FL robustness by selecting the mean and median value of each gradient parameter; these coordinates are subsequently organized as the aggregation outcome. To avoid the aggregation outcome being impacted by the extreme value, Trimmed mean first discards a preset percentage of extremum. Bulyan [45] combines the Krum and Trimmed-mean algorithms. Bulyan first selects a number of gradients as candidates by repeatedly performing Krum. and removes them from the candidate pool. Then, Bulyan performs Trimmed-mean among the candidates and generates the outcome to update the global model.

However, these previously mentioned methods operate under the strong assumption that the defender is aware of the adversary's volume and distribution, an assumption that may be challenging to apply in practical scenarios. Research [15] addresses this limitation by proposing the Sniper scheme. Sniper [15] does not require prior knowledge of the number of malicious users and identifies benign local model updates by solving the Maximum Clique Problem (MCP). Similarly, study [102] relies on the Markov model with median and cosine similarly to discard the suspicious local model updates in each iteration and does not require the knowledge of the adversary either.

B) Local Model Performance-Based Robust Aggregation

To evaluate the honesty of clients' model updates and limit the participation of suspicion, some recent studies introduce an evaluation dataset to decide the aggregation weight of each client. For instance, FL Trust [16] proposes a novel solution by keeping a server-version

model. Under FL Trust, the server first collects a small and clean dataset as the training dataset. Then, the server normally keeps and trains the model (called the server-version model) in each iteration. Once the clients' gradients are received, the server calculates the cosine similarity between them and the server-version gradient; the gradient achieving high cosine similarity will subsequently gain a high weight in aggregation. On the other hand, FL Trust normalizes the magnitudes of clients' gradients through the server-version model update to defend against scaling attacks. Similarly, Sageflow [110] evaluates the entropy of each gradient received through the evaluation dataset and subsequently divides the weight to different clients based on the entropy generated. Sageflow also considers the volume of training data in each client; the client owning a large data size can gain a large weight in the aggregation and vice versa. Research [155] proposes a score-based ranking mechanism, namely Zeno. Zeno suspects clients that are potentially defective, which is tolerant to an arbitrary number of attackers with at least one benign client. Based on Zeno, Zeno++ [156] has been introduced to further mitigate the communication bottleneck, allowing for asynchronous communication. Research [19] generates the gradient of noise based on a small clean dataset. The noise gradient is subsequently compared with the clients' model updates to identify and filter out suspicious gradients.

Although the existing byzantine-robust aggregation can achieve robustness under particular assumptions like IID distribution of overall training data, the knowledge of clients' data distribution, or a set of clean public data, these assumptions can not be guaranteed in the practical FL scenarios, which usually results in degrading defense performance and leaves opportunities for state-of-the-art attacks to poison the model.

C) Training Function Optimization-Based Robust Aggregation

Different from the aforementioned aggregation rules that filter or limit the suspicious based on their gradient, training function optimization-based robust aggregation leverages the DL loss function optimization to achieve byzantine resilience. Compared with other robust aggregations, training function optimization-based robust aggregation is still in the early research stage. Research [83] introduces regularization in the loss function to prevent the local model from deviating excessively from the global model during training. Instead of

relying on a fixed clipping norm, study [3] proposes a method with dynamic clip value at the specific quantile estimated online.

2.4 Fairness-Enhanced FL Methods

Although the server acts as a controller in the traditional FL setting, which should not make unfair decisions to any particular clients or data owners, fair treatment for the participants has not been highlighted, and fairness problems could arise during the while FL training process. For instance, fairness issues may occur during client selection of initialization [57, 159, 129], model optimization in aggregation [101, 87], contribution evaluation in each iteration [100, 66, 164] and incentive distribution after finishing the training task [172, 139]. Today, many methods aimed at augmenting fairness in FL have been introduced. We survey these works in this section, categorizing them based on the different stages of FL.

2.4.1 Client Selection

In FL algorithms, the server could first select the client to participate in the learning task from its interest. As a result, the disadvantaged clients may hardly participate and be included in the learning process, receiving the global model that does not fit their local data distribution or non-incentives. To mitigate the bias and unfairness, some recent works propose sampling constraints for FL client selection. [57] proposes long-term fairness constraints for enhancing FL client selection fairness. By leveraging the proposed constant fairness parameter, each client can achieve a preset participation rate. Similarly, research [159] also focuses on promoting the participation of the clients being disadvantaged. Specifically, [159] introduces the Combinatorial Multi-Armed Bandit (CMAB) to formulate the FL client selection problem; each client is represented by the arm, and its incentive is defined by the raw data's class distribution. Study [129] takes the real-time contribution of individual participants into account by proposing a reputation-based client selection scheme. The proposed algorithm keeps a reputation table to maintain the clients' historical behavior; a fairness parameter is further introduced to balance the reputation and number of successful transmissions. This

fairness parameter limits the high reputation of clients achieving a large number of successful transmissions and enables the weak clients to participate in the learning task.

The client selection scheme could also be designed from the client model customizing perspective. Study [7] proposes the Federated Dropout (FD), which enables each participant to receive a suitable size sub-model based on the difference in computational capabilities. For clients achieving disadvantaged computing resources, the activations and a preset percentage of filters are usually dropped out from the full connection and conventional layers, respectively. Similarly, [12] proposes Adaptive Federated Learning (AFD) to promote the fair treatment of different clients and allows the clients owing low capabilities to participate in the learning task. AFD keeps a map, called the activation score map, to select the important activation notes and generates the best-customized model for each client. Besides the customized model for heterogeneous clients, some existing works also consider customizing the workloads for FL clients to enhance fair treatment. A recent work, FedProx [86], enables clients to set different local training epochs instead of assigning fixed and uniform epochs.

2.4.2 Model Optimization

The fairness issue may also arise during the model optimization stage of FL. This could potentially result in the global model displaying discrimination against a particular client group, thereby causing disparate performance levels among clients. Recent works [101, 54, 28] investigate this fairness issue and propose solutions from two perspectives: objective function-based solutions and gradient-based solutions.

Objective function-based solutions pertain to an approach that seeks to optimize the global or local objectives of the FL model while simultaneously adhering to the target fairness constraints. Through these methods, different clients can achieve and maintain a similar average accuracy. AFL [101] is one of the most representative of Objective function-based solutions; it aims to achieve the good-intent notion of fairness by avoiding the global model overfitting on any particular clients. AFL optimizes the global model for any target distribution by the mixture of some participants and does not degrade the performance of the others.

Similarly, q-FFL [87] has been introduced to address the FL fairness issue but extends the capabilities of the AFL to accommodate a larger number of participants. Drawing inspiration from resource allocation in wireless networking, q-FFL introduces a parameter, q, to reweigh the aggregate loss. This parameter assigns weights based on individual loss values; clients achieving high loss will receive high weights and vice versa.

Recent studies have explored the use of multi-objective optimization to reduce the variance of model performance across participants. FedMGDA+ [54] conducts multi-objective optimization by independently and concurrently optimizing the loss function for each FL client to enhance both FL model fairness and robustness. It employs Paretostationary solutions to identify a unified descent direction for all chosen clients, avoiding degrading any client's performance. Additionally, FedMGDA+ [54] leverages two techniques, namely gradient normalization and a Chebyshev-inspired build-in robustness method, to defend against the model degrading attacks. Another study [28] employs multi-objective optimization to attain group fairness, termed FCFL [28]. Instead of relying on the single non-smooth objective associated with the poorest-performing participant, this study utilizes a smooth surrogate maximum function that takes into account the objectives of all participants.

Different from the aforementioned solutions that emphasize fairness at the individual client level, the authors of AgnosticFair [33] consider examining and improving fairness at the group level by employing kernel re-weighting functions. Thanks to the agnostic fairness constraint, AgnosticFair exhibits good performance in situations with unknown data distributions and data shifts. A recent work, FairFed [36], enhances group fairness through a novel algorithm for fairness-aware aggregation. The proposed algorithm is server-side, enabling the flexible adoption of various local debiasing methods across clients and consequently ensuring group fairness amidst data heterogeneity.

Gradient-based approaches guarantee uniform performance across different clients by fairly aggregating the clients' gradients; the gradient mentioned here indicates the model updates in each learning round. From a broader perspective, gradient-based solutions are still in their early stages of development. FedFV [148] is introduced to resolve conflicts among client gradients before the gradient averaging in the global objective function. Specifically, FedFV

[148] uses gradient projection to mitigate the conflicts from both internal and external, which refer to the conflict among participants and between participants and unselected candidates, respectively. Specifically, FedFV [148] employs gradient projection to alleviate conflicts both internally and externally, which pertains to conflicts among participants and conflicts between participants and unselected candidates. While FedFV demonstrates effectiveness when gradient projection is accurate, its reliability may diminish when estimations become out-of-date.

2.4.3 Contribution Evaluation

The contribution evaluation process enables the FL server to assess the importance of the contribution carried out by each participant and subsequently provide incentives. Without a fair contribution evaluation process, the participants may not engaged to join the FL task and provide local resources [65].

A fair contribution evaluation could be performed based on clients' self-reported information, including data quality, quantity or computation, and communication cost. For instance, research [100] asks the clients to report publicly verifiable factors of their data and subsequently assigns a ranking to these clients. Similarly, research [66, 164] designs the incentive mechanism based on clients' reports and Contract Theory [11]. The clients can select the most appealing contract as the commitment to join FL tasks. Drawing inspiration from the Stackelberg Game [127], study [122] introduces a novel FL incentive scheme. Through self-reporting information, each participant can submit their desired price for dedicating a specific unit of computing resource and other associated costs frequently. Because of the privacy-preserving character, the FL server can not directly access the local device and identify the truth of the report, which leaves opportunities for dishonest/untrusted clients and brings challenges for deploying in practice.

Designing contribution evaluation approaches through Shapely Value (SV) [124] is also an appealing research trend. Research [128] introduces one-round reconstruction (OR) and multi-round reconstruction (MR), two SV methods. On the one hand, OR gathers all gradients

during the training and reconstructs the models for all subsets in the final iteration with SV calculation. On the other hand, MR computes a set of SVs each round and subsequently calculates the final SV-based contribution values. Truncated multi-round (TMR) [150] builds upon MR [128] by allocating higher weights to learning rounds with high accuracy and by eliminating superfluous model reconstruction. Research [144] proposes a feature importance evaluation approach in VFL scenarios. Instead of calculating SV for each individual feature, this study calculates the SV value on groups of features to prevent the disclosure of sensitive features and information.

Some recent studies [66, 67] consider keeping a record of each FL participant and evaluating their contribution based on their historical reputation. The reputation record could consist of both direct and indirect reputation opinions collected from this or other service providers. For instance, research [66] investigates recording reputation through an attack-detecting mechanism. Study [67] leverage computation time and local data size for evaluating reputation. However, most of these works assume clients are trustworthy, which may not be realistic when deploying in the real world.

2.4.4 Incentive Mechanisms

Participants might be reluctant to engage in the FL task and contribute their training data and computing resources if the server does not offer a fair incentive distribution [65]. To mitigate this research gap, many FL incentive distribution mechanisms have been introduced in recent years. Broadly, these can be categorized into monetary incentive mechanisms and non-monetary mechanisms.

The monetary incentive indicates the server (or service provider) distributes monetary pay-off to FL task participants. For instance, study [172] proposes FMore to engage the clients owing high-quality data to join the training task by introducing the game theory. FMore starts from a bidding stage, during which it broadcasts its asking price with the scoring rule to the clients. The scoring rule is shown below:

$$S(q_{i1}, q_{i2}, ..., q_{im}, p_i) = s(q_{i1}, q_{i2}, ..., q_{im}) - p_i \tag{2.1}$$

Here, $s(\cdot)$ denotes the utility function of the server, and the vector $q(\cdot)$ indicates the quality of each client like data capacity, computing resource, network, etc. p_i represents the expected payment of client i. Following the bid-asking stage, each client can decide to participate in the bidding. After collecting a predetermined number of bids, the server selects participants based on their anticipated contribution. Ψ-FMore [172] extends FMore by introducing a probability parameter Ψ, which enables the client to achieve low recourse can participate in the learning task.

To ascertain the resource cost information from participants, recent studies consider asking clients to submit reports detailing both costs and data quality. Fair-VCG (FVCG) [139] has been proposed based on the VCG mechanism, which assigns rewards to clients by the self-submitted report. However, given that the veracity of self-submitted reports cannot be assured, the Fair-VCG (FVCG) faces challenges when implemented in practical settings. To mitigate the research gap, study [175] proposes RRAFL based on a reputation mechanism. Specifically, RRAFL monitors clients' historical behaviors to determine a direct reputation value rather than depending on unverified reports; the historical records could be collected or shared from other FL servers. The task publisher first ranks the participants based on the reputation bid prices and selects the last k participants, the $k+1$-th participant's unit bid price is used to determine the pay-off of all participants.

Contract theory [11] is an emerging and compelling research direction for designing FL incentive mechanisms. By leveraging on contract theory, study [66] proposes a novel incentive algorithm for attracting clients with high-quality local data to participate in the FL task. A parameter of local data quality is defined as the type of contract model, which enables the server to customize the contracts for clients achieving different data quality. Consequently, participants owning advantage resources, whether in computing or training data, can receive higher rewards from the server in [66]. Similarly, research [164] introduces the contract theory and designs a novel incentive mechanism in FL vehicular edge computing scenarios. Study [164] proposes a 2-D contract to assign awards to each client according to their computing resource and data quality. While these contract theory-based incentive mechanisms enhance

the fairness of FL by ensuring appropriate rewards, they follow the assumption of monopolist (a single FL task publisher), which may not align with real-world FL settings.

As the monetary incentive is unavailable in some settings, recent works also design incentive mechanisms from a non-monetary perspective. Study [100] considers rewarding the client by contributing high-quality training data with high-quality model updates and proposes a hierarchically fair FL (HFFL) framework. Based on the characteristics of each client, HFFL first classifies them into different levels and consequently trains one model for each level. When training the lower-level models, the higher-level clients contribute an equivalent amount of data to those clients at the lower level. As a result, the high-level client may receive an FL with better performance due to training with a larger and higher-quality dataset. However, since HFFL relies on self-submitted reports, it faces the same challenges discussed in client selection when participants are not truthful.

Instead of assigning the participants into different groups [100], study [94] proposes a decentralized framework, namely, fair and privacy-preserving deep learning (FPPDL). Under FPPDL, each client can receive a number of transaction points based on its credibility and commitment level. These points can subsequently be used to download gradients from other participants. As a result, in FPPDL, each client receives a different variant of the global model according to its contribution.

CHAPTER 3

Reducing FL Attack Surface

In this chapter, we present our solution to reduce the large attack surface of FL inherent in non-IID settings. We identify the main non-IID resource of FL and analyze how to achieve the attack surface reduction by leveraging the insight. Our proposed framework, namely Mini FL, dynamically assigns the client to different groups based on a predefined grouping principle and aggregates the gradients respectively. To demonstrate the performance of Mini FL, a real-world case study and comprehensive experiments are performed, where the results show a significant improvement of Mini FL compared to the benchmarks. Our MiniFL is an extension of our previous work, Enhancing Federated Learning Robustness Through Clustering non-IID Features, which was published at the 2022 Asian Conference on Computer Vision (ACCV). Our Mini FL framework is published at Computers & Security under the name Enhancing Federated Learning Robustness in Adversarial Environment through Clustering Non-IID Features.

3.1 Introduction

Federated learning (FL) enables many clients to train a machine learning model collaboratively [65], but the distributed nature of training data makes FL vulnerable to various attacks (such as poisoning attacks) by malicious attackers and untrusted clients. Poisoning attack [61, 63, 137], which seeks to damage the model and generate misbehavior, draws the most important threats to FL security. To defend against poisoning attacks, byzantine-robust aggregation rules [9, 45, 16] have been introduced on the server to update the global model. By comparing the client's model updates, these aggregation rules can limit the attack surface, discard the

statistical outliers, and prevent the suspected model uploaded from poisoning the global model. Although most of the studies [9, 167, 123] are designed and evaluated in an Independent and Identically Distributed (IID) setting and assume each client's data follows the same probability distribution, the training data in real-world FL applications are usually non-IID, making the existing byzantine-robust FL methods show little effectiveness and even fully break when facing state-of-the-art attacks [37].

The most common sources of non-IID are a client corresponding to a particular location [5, 95, 52, 102, 65], a particular time window [62, 1, 118, 113, 46], and/or a particular user cluster [65, 40, 77, 114]. In terms of location, various kinds of location factors drive the most impact on the non-IID of a dataset. For instance, the mammal's distributions are different due to the geographic location [52], customer profiles are different due to various city locations [102], and emoji usage patterns are different due to the demographic locations [65]. In terms of a time window, people's behavior and objects' features can be very different at different times. For instance, the images of parked cars are sometimes snow-covered due to the seasonal effects, and people's shopping patterns are different due to fashion and design trends. In terms of a particular user, different personal preferences can result in a dataset non-IID. For instance, [26] shows students from different disciplines and schools have very different library usage patterns.

In this section, we first evaluate the effectiveness of the existing byzantine-robust FL methods in different level non-IID settings. Our findings indicate a decline in performance for these methods as the degree of non-IID increases. We note that benign client gradients exhibit a high similarity in the IID setting, which leads to a small attack surface, allowing the existing FL methods to successfully identify manipulated gradients, even when minor perturbations are introduced. In contrast, clients exhibit heterogeneous behavior under non-IID settings. This heterogeneity results in a substantial increase in the attack surface, which allows crafted gradients to blend in with the normal gradients, posing a challenge for byzantine-resilient FL methods to identify and discard suspicious gradients effectively. To reduce the attack surface and enhance the robustness of FL in non-IID scenarios, we propose a new FL framework, namely the Mini-FL framework. The rationale of Mini FL is that the attack surface can be

effectively reduced by identifying non-IID resources and setting a grouping principle to assign clients' gradients accordingly. Figure 3.1 illustrates the attack surface of the existing FL and proposed Mini FL methods in IID and non-IID settings.

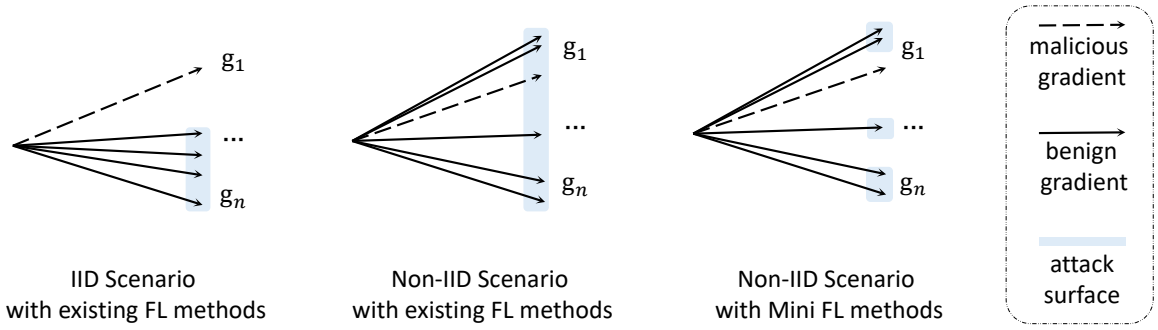

FIGURE 3.1: The illustration of the attack surface of the existing FL and our proposed Mini FL methods under IID and non-IID settings.

In particular, Mini-FL considers the main source of non-IID and identifies Geo-feature, Time-feature, and User-feature as the alternative grouping features. Based on the grouping feature selected, the server defines the grouping principle by performing unsupervised learning. In each iteration, the server first assigns the received gradients to different groups and then performs byzantine-robust aggregation, respectively. Finally, the server aggregates the aggregation outcomes (called group gradient) from each group to update the global model in each iteration. We use Krum [9], Median [167], and Trimmed-mean [167] as the byzantine-robust aggregation rule to evaluate our Mini-FL on the various dataset from different non-IID levels. Our results show that Mini-FL effectively enhances the robustness of existing byzantine-robust aggregation rules and also reaches a high level of accuracy (without attack) in the extreme non-IID setting. We further provide a case study to further demonstrate the effectiveness and practicality of Mini-FL in the real world.

3.2 Related Work

3.2.1 Poisoning Attacks on Federated Learning

Poisoning attacks generally indicate the attack type that crafts and injects the model during training time. These attacks include data poisoning attacks [8] and model poisoning attacks [37, 52, 102, 99, 29], which are performed by poisoning the training data owned and gradients, respectively. The model poisoning attack directly manipulates gradients, which can bring higher attack impacts to FL. Based on the adversary's goals, the attacks can be further classified into untargeted attacks [37, 52, 102, 99, 29] (model downgrade attacks) and targeted attacks [42, 35] (backdoor attacks). In untargeted attacks, the adversary aims to reduce the global model's accuracy and entirely "break" the model by participating in the learning task. In contrast, target attacks maintain the global model's overall accuracy but insert "back door" in minority examples. These back-doors can result in a wrong reaction when the attacker-chosen action event occurs. As the untargeted draws lead to security threats for FL, we consider the setting of untargeted model poisoning attacks.

On the one hand, some model poisoning attacks rely on a fixed attacking strategy to perturb the local model update. For instance, Reverse attack [29] and Random attack [99] craft the gradients by reversing the original local model update and replacing the benign gradient parameters with random values. The Partial Drop attack drops the gradient parameter with a given probability and consequently fills the vacancies with a 0 to generate the malicious gradient. However, since these attacks have not taken into account the benign clients, the crafted gradients often result in a significant bias compared to the benign gradients, which can be effectively detected by the byzantine-robust FL methods. To make the malicious gradient effectively hide behind the benign, some state-of-the-art attacks leverage the difference that exists between normal clients and the dimension curse of machine learning. For instance, Little is Enough attack [6] and Fall of Empires attack [154] upload the crafted gradient by adding perturbation on the mean of the gradient owned (based on the capability). Local model poisoning attack [37] first infers the convergence direction of the gradients and uploads the scaled, reverse gradient to poison the global model.

3.2.2 Byzantine-Robust Aggregation Rules

The FL server can effectively average and aggregate the local models received in non-adversarial settings [97, 106, 151]. However, linear combination rules, including averaging, are not byzantine resilient. In particular, a single malicious worker can corrupt the global model and even prevent global model convergence [9]. Therefore, the existing byzantine-robust aggregation rules have been designed to replace the averaging aggregation and address byzantine failures. Here, we discuss several representative byzantine-robust aggregation rules and the corresponding FL methods.

Krum [9] discards the gradients that are too far away from benign gradients and considers the client closest to others to be the most honest. In particular, for each gradient received, Krum calculates the sum Euclidean distance of a number of the closest neighbors as the score. The gradient with the lowest score is the aggregation outcome and becomes the new global model in this iteration. As the number of the closest neighbors selected influences the score, Krum requires the number of attackers. Trimmed-mean[167] is a coordinate-wise aggregation rule which aggregates each model parameter, respectively. For a given parameter, the server firstly sorts the parameter from all gradients received. Then, the server discards a part of the largest and smallest values and finally averages the remaining gradients as the corresponding parameter of the new global model in this iteration. The Median [167] method is another coordinate-wise aggregation rule. In the Median method, the server firstly sorts the parameter from all gradients received and selects the median as the corresponding parameter of the new global model in this iteration.

We compare the robustness of several FL methods and their corresponding Mini-FL methods against different attacks under the IID and non-IID settings in Table 3.1. We use the accuracy of the Standard FL method (with Vanilla aggregation) in the non-adversarial setting as the baseline to compare with. Here, "$\checkmark$(effective)" denotes the global model can maintain a similar accuracy, and "$\times$ (ineffective)" denotes the global model has been fully broken. "O (partially effective)" denotes the target FL method can maintain robustness only in some particular, moderate non-IID scenarios but drops the global accuracy when facing state-of-art

attacks or high non-IID degrees. We note our Mini-FL framework can collect the information of each client cluster and effectively enhance the robustness in non-IID settings; the full evaluation information of different FL and Mini-FL methods against adversaries is shown in Section 3.6.

TABLE 3.1: Illustration of the robustness of the existing FL and Mini-FL methods against different attacks under the IID and non-IID settings.

Attack	**Reverse, Random**		**Partial Drop**		**Little, Fall**		**Local Poisoning**	
Data Distribution	IID	non-IID	IID	non-IID	IID	non-IID	IID	non-IID
Vanilla	×	×	×	×	×	×	×	×
Krum	✓	×	✓	×	O	×	×	×
Trimmed Mean	O	×	×	×	O	×	O	O
Median	✓	O	✓	O	✓	O	O	×
Mini Krum	✓	✓	✓	✓	✓	✓	✓	✓
Mini Trimmed Mean	✓	O	✓	×	✓	✓	✓	×
Mini Median	✓	✓	✓	✓	✓	✓	✓	✓

3.2.3 Clustering Federated Learning

In the federated learning task, the training data is considered distributed in a non-IID fashion because clients may behave heterogeneously across different IPs, time windows, and client clusters. To identify these clusters and further benefit FL, clustering methods have been widely introduced in recent FL research [14, 79, 123, 40, 72]. Here, we discuss several popular clustering federated learning algorithms; we note that most of these studies have considered different scenarios from our work. For instance, [14, 79, 40] consider a benign learning environment, [123] consider an IID scenario, while our work setup in the non-IID and adversarial scenario.

Federated Learning + Hierarchical Clustering (FL+HC) [14] introduces a hierarchical clustering step in the standard FL algorithm to enable a more significant percentage of clients can reach the target accuracy compared to standard FL. Specifically, the clustering step (HC) is introduced at a preset round to merge the most similar clusters of clients based on their local

model updates. Then, the clients in each determined cluster are trained independently but simultaneously with the initialization of the current joint model. Considering the client may belong to multiple clusters in real-world scenarios, Federated Learning With Soft Clustering (FLSC) [79] introduces a soft clustering method to capture the complex nature of real-world data. Within FLSC, clients are assigned into overlapping clusters, and the information of each participant can be utilized by multiple clusters concurrently with each iteration.

Our work most closely resembles that of Clustered Federated Learning (CFL)[123], which presents the CFL algorithm to enhance the robustness of FL in IID settings. CFL first calculates the cosine similarity between each client's gradient and merges them if under a preset threshold. The gradients within the largest cluster are regarded as benign and consequently participate in the aggregation; all other gradients are discarded.

3.3 Problem Setup

3.3.1 Threat Model

We consider the adversary controls some clients and aims to reduce the model's global accuracy through untargeted model poisoning attacks [65, 42, 35]. The attackers can access the dataset on the controlled device, which is owned by the original client, and utilize the dataset to generate the crafted gradient. We assume the attackers can collude in each iteration and know the aggregation rules. Although these assumptions maximize the attack performance, we demonstrate our Mini-FL framework can achieve byzantine robustness even against attackers with strong capability in Section 3.6. Specifically, the attacking processes are shown as follows: The attackers first agree on an attack strategy based on the current learning task. After receiving the current global model from the server in each iteration, the attacker generates the crafted model updates based on the data recourse owned and the poisoning strategy selected. Then, the crafted gradients are sent back to the server to poison the global model. The adversaries will repeat the poisoning process during the learning task.

We consider the adversary has no knowledge about whether it has been included or discarded from the current aggregation.

As our Mini-FL framework should work on an existing FL framework, we keep the setting of the adversary amount's upper boundary of each existing FL method. For example, the Mini-Krum keeps the attacker upper bound as $f < (n-2)/2$, which is the same as the original Krum method. Table 3.2 illustrates the attacker upper bound of the FL methods introduced in the Mini-FL framework in this paper, where f denotes the number of attackers and n denotes the amount of all learning participants in the iteration. As the Vanilla aggregation method is not robust (attacker upper bound is 0), it has not been selected by our Mini-FL frameworks and is listed in the table.

In Mini-FL, the grouping principle used to partition the clients' gradients is entirely defined by the server and could be updated on demand (full information is given in Section 3.4.2). We note that the adversary can know the grouping principle only if the attacker keeps controlling the server during the learning task, which is not practical. Hence, in this study, we assume the adversary does not know the grouping principle.

TABLE 3.2: Illustration of the adversary amount upper bound of different FL and proposed Mini-FL methods; f denotes the adversary amount, and n denotes the number of overall participants.

Aggregation Rule	Mini Krum/Krum	Mini T Mean/T Mean	Mini Median/Median
Attacker Upper Bound	2f+2 <n	2f<n	2f<n

3.3.2 Defender Profile

The defense objective includes two parts; on the one hand, the proposed FL framework should achieve byzantine robustness against untargeted attacks in non-IID scenarios. On the other hand, the framework should embody the data minimization principle. Specifically, the new framework does not need clients to upload further information beyond local model updates.

The server plays the defender's role and has access to the information naturally brought with the gradients uploaded (e.g., IP, Timestamp, etc.). We notice some byzantine-robust aggregation rules need to know the upper bound of the malicious clients [9] [167]; we follow these settings but don't leak further information about malicious clients; specifically, the defender does not know the distribution of malicious clients.

3.4 Mini-FL Design and Analysis

3.4.1 Overview of Mini-FL

In our Mini-FL, the server assigns the model updates received into different groups and executes byzantine-robust aggregation accordingly. Specifically, Mini-FL follows the general FL framework but introduces a new step (i.e., Grouped Model Aggregation) before the Global Model Update. Furthermore, a Prepossessing Step (i.e., Grouping Principle Definition) is introduced before the training task starts. Figure 3.2 demonstrates the overview of the Mini-FL framework.

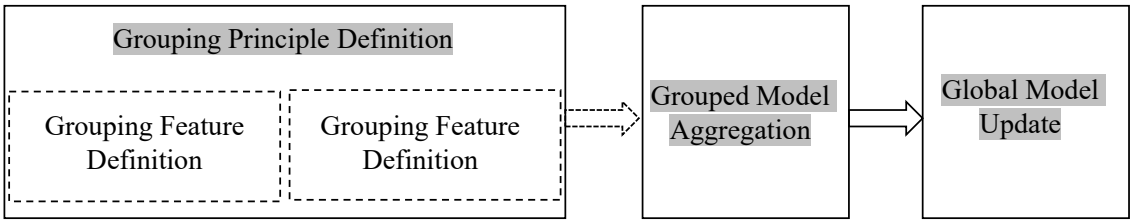

FIGURE 3.2: The overview of the Mini-FL framework.

To craft the malicious gradient and avoid being excluded by byzantine-robust aggregation rules, the adversary commonly statistically analyzes the gradient owned and calculates (or infers) the range of the benign gradients. By restricting the crafted gradient under this range, the attackers can effectively hide their gradients in benign gradients and subsequently attack the global model. However, because most federated learning models are trained through non-IID data, the gradients uploaded naturally tend to be clustered due to location, time, and user cluster reasons. Thus, Mini-FL first defines the groups and then executes byzantine-robust

aggregation accordingly. The similar behavior of each group brings a smaller gradient range and, therefore, results in a smaller attack space. Finally, the server aggregates the outcome from each group and updates the global model to finish the current iteration.

3.4.2 Mini-FL Framework

Our Mini-FL considers and leverages the non-IID nature of federated learning to define groups of the learning task and execute byzantine-robust aggregation accordingly. Figure 3.3 illustrates the Mini-FL framework. Specifically, Mini-FL includes the following steps:

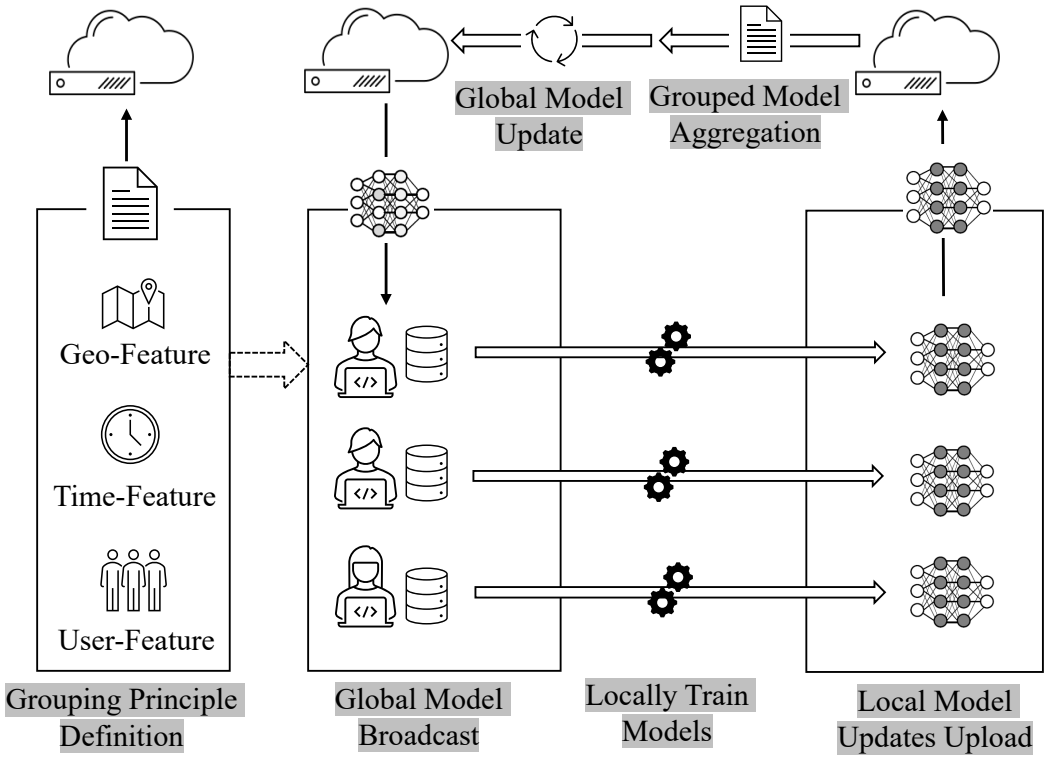

FIGURE 3.3: Illustration of the Mini-FL framework.

A) Grouping Principle Definition

Before the learning task starts, the server defines the grouping principle (i.e., Prepossessing Step), which includes "Grouping Feature Definition" and "Grouping Boundaries Definition";

the grouping principle could only be defined before the learning task starts or is required to be updated.

- Grouping Feature Definition: The existing research [65] believes the major sources of non-IID are due to each client corresponding to a particular geographic location, a particular time window, and/or a particular user. For instance, [52] demonstrates the real-world example of skewed label partitions: geographical distribution of mammal pictures on Flickr, [65] illustrates the same label can also look very different at different times(e.g., seasonal effects, fashion trends, etc.).

  Considering the major source of non-IID and the features naturally carried in server-client communication, we identify the Geo-feature (e.g., IP address), Time-feature (e.g., Timestamp), and User-feature (e.g., User ID) of the local model update as the based grouping feature to maintain the principle of focused collection and guarantee the effectiveness of clustering.

  When defining the grouping feature, the server firstly regroups the gradient collection C by Geo-feature; the collection C should accumulate the gradients received in a few iterations to maintain the generality. Then, we execute the "Elbow Method"[73] to detect the number for clustering and subsequently get the SSE (i.e., Sum of the Squared Errors, which reflected the grouping effectiveness). By repeating the first two steps through replacing the Geo-feature with Time-feature and User-feature, we can find the feature F with the lowest SSE. Finally, we select that feature F acts as the grouping feature and the corresponding elbow point as the number of groups.
- Grouping Boundaries Definition: Once the grouping feature has been defined, we cluster the collection regrouped through unsupervised learning. In this research, we use the K-means[25] algorithm to execute the unsupervised learning, and we use the "elbow" point under the selected feature F (which is generated in the Grouping Feature Definition step) as the number of groups (i.e., K) for clustering. By analyzing the gradient's feature value in different groups, the grouping boundaries could be defined.

In the Grouping Principle Definition step, all operations (i.e., elbow method and K-means method) are performed based on the gradients sent from clients. Compared with the standard FL, The only additional information the Mini-FL used is/are the Geo-feature, Time-feature, or/and User-feature. We select these three features as they are the main source of non-IID in FL, and the gradients naturally carry them during the most general server-client communication process. In other words, these features can be available to be collected even if they are not recorded in the client's data (consider the communication process when sending gradient to the server, the IP address of the message source, the response time, or the time stamp, and the device ID are naturally carried by the message (gradient)), which does not break the client's privacy.

B) Grouped Model Aggregation

According to the grouping principle, the server divides the gradients received into different groups and executes byzantine-robust aggregation, respectively. The Mini-FL framework has strong generality and can utilize most existing byzantine-robust aggregation rules. In this research, we use Krum, Trimmed-mean, and Median for aggregation in this research, and the details of the experiments are studied in Section 3.6.

C) Global Model Update

The server calculates the weighted mean of grouped gradients (i.e., the aggregation outcome from each group) based on the volume of each group to generate the global gradient and updates the global model to finish this learning iteration.

3.4.3 Toy Example

Here, we provide a toy example to illustrate further and demonstrate the Mini-FL working process. Suppose a learning task upgrades the existing FL framework to the Mini-FL framework, and the server receives m local model updates. We assume the server has kept a historical recording of the client's gradients and associated information.

A) Processing Step

Server records m local model updates with nature information carried by the updates. Each

update item is assigned an ID with its Geo-feature (IP address), Time-feature (sending time), and User-feature (Client ID); Table 3.3 illustrates the detail of the records.

TABLE 3.3: The records of revived local model updates and associated information.

Update ID	Geo-Feature	Time-Feature	User-Feature	Updates
U1	IP2	Time1	C1	G1
U2	IP1	Time1	C2	G2
U3	IP3	Time2	C3	G3
U4	IP2	Time1	C2	G4
U5	IP3	Time3	C4	G5
		...		
Um	IP4	Time3	C4	Gm

TABLE 3.4: Illustration of the gradient records regrouped.

(a) Recording regroup by Geo-feature					
IP address	IP1	IP2	IP3	IP4	...
Local model updates	G2	G1,G4	G3,G5	Gm	...
(b) Recording regroup by Time-feature					
Time Stamp	Time1	Time2	Time3	Time4	...
Local model updates	G1,G2,G4	G3	G5,Gm	...	...
(c) Recording regroup by User-feature					
Client ID	C1	C2	C3	C4	...
Local model updates	G1	G2,G4	G3	G5,Gm	...

- Grouping Feature Definition: The server uses Geo-feature (IP address) as the index and regroups the local model updates; Table 3.4 illustrates the regrouped records. Once the regrouped records are generated, the server performs the "Elbow Method" for new records and subsequently gets the SSE_{Geo}. Then, the server replaces the Geo-feature (IP address) with the Time-feature (Time) and User-feature (Client ID) and generates the corresponding SSE_{Time} and SSE_{User}, respectively. Suppose $SSE_{Geo} < SSE_{User} < SSE_{Time}$; then the Geo-feature becomes the grouping

feature in the whole learning task until the training data pattern has changed (which may drive influence on the non-IID source of the target learning task).

- Grouping Boundaries Definition: The server performs the K-means algorithm for the regrouped records; suppose the results are generated as Table 3.5.

TABLE 3.5: Illustration of the K-means result.

(a) Geo-feature acts as the grouping feature				
Group	Group1	Group2	Group3	/
Geo-feature	IP1	IP2, IP4	IP3	/
(b) Time-feature acts as the grouping feature				
Group	Group1	Group2	Group3	Group4
Time-feature	Time1	Time2	Time3	Time4
(c) User-feature acts as the grouping feature				
Group	Group1	Group2	Group3	/
User-feature	C1,C2	C3	C4	/

So far, the grouping principle has been determined. Specifically, the updates that IP belongs to IP 1 range is assigned to $group1$, IP belongs to IP 2 or IP 4 range is assigned to $group2$, and IP belongs to IP 3 range is assigned to $group3$.

In this proposed example, the Geo-feature achieves the lowest SE and consequently acts as the grouping feature; the corresponding IP addresses clustered define the grouping boundaries (Table 3.5 (a)). We also list the grouping boundaries if Time-feature and User-feature act as the grouping feature in Table 3.5 (b) and 3.5 (c); we note that there are different amounts of groups defined under different features.

B) Global Model Aggregation

In each further iteration, the server assigns the updates to $group1$, $group2$, and $group3$ based on its IP address and the grouping principle defined. Suppose we use Krum [9] as the aggregation rule, $group1$, $group2$, and $group3$ perform Krum aggregation respectively and subsequently generate the grouped gradients: G_{group1}, G_{group2}, and G_{group3}.

C) Global Model Update

The server proportionally averages G_{group1}, G_{group2}, and G_{group3}, suppose $\frac{1}{4}$ gradients received are assigned to $group1$ and $group3$, $\frac{1}{2}$ gradients received are assigned to $group2$; the global aggregation result G_{aggre} is:

$$G_{aggre} = \frac{1}{4}G_{group1} + \frac{1}{2}G_{group2} + \frac{1}{4}G_{group3}$$

Then, the server uses G_{aggre} to update the global model and finish this iteration.

3.5 Security Enhancement Analysis

Here, we analyze the security enhancement of Mini-FL from Information Asymmetry and Attack Surface Reduction two perspectives.

3.5.1 Information Asymmetry

As discussed in Section 3.2, most existing byzantine-robust aggregation rules can effectively detect and discard the malicious gradient if it is far (based on Euclidean distance) from benign gradients. To guarantee the effectiveness of attacks and avoid being excluded by the byzantine-robust aggregation rules, a common perturbation strategy introduced in the state-of-the-art attack strategies is determining the attack direction and then scaling the crafted gradient to stay close to benign gradients. Depending on different knowledge, the adversary can precisely or generally infer the statistics (e.g., max, min, mean, and standard deviation (Std)) of the benign gradients and subsequently scale the crafted gradient; Table 3.6 illustrates the scalier of gradient crafted in different attacks.

In Mini-FL frameworks, all grouping principle definitions and client gradients distribution processes are performed only on the server side; all clients, including the adversary, have no knowledge about the grouping information. The information asymmetry makes the adversary hardly infer the members of different groups, much less calculate the relevant statistical parameters to scale the crafted gradients and bypass the defense of Mini-FL.

TABLE 3.6: The crafted gradients range under different poisoning attacks.

Poisoning Attacks	Crafted Gradients Range
Little is enough Attack	$(\mu - z\sigma, \mu + z\sigma)$ μ:mean, z:scalar (set 0 $\sim$ 1.5 in research), σ:Std.
Fall of empires Attack	$(-z\mu, -z\mu)$ μ:mean, z:scalar (set 0 $\sim$ 10 in research), σ:Std.
Local model poisoning Attack with partial knowledge	$(\mu + 3\sigma, \mu + 4\sigma)$or ($\mu$ - 4σ, μ - 3σ)
Local model poisoning Attack with full knowledge	$(W_{max}, z * W_{max})$ or $(z * W_{min}, W_{min})$ The direction depends on the global gradient direction. μ:mean, z:scalar (set 2 in research), σ:Std, W_{max}/W_{min}:the max/min gradient value at that iteration.

3.5.2 Attack Surface Reduction

Most existing byzantine-robust aggregation rules organize the attack surface by evaluating the Euclidean distances of the benign gradients; the attacker can craft its gradient to stay close to the boundary of the attack surface to perform the attack. In other words, as any gradient beyond the boundary would be discarded, the attack surface limits the max of the perturbation.

Here, We formally derive the reduction of the attack surface by Mini-Krum, comparing with the Krum method[9] when defending against Local Model Poisoning attack [37]. Similar derivation can be applied to other defense and attack models. We first show that the attack surface is only limited by the benign models and then demonstrate the reduction of the attack surface by Mini-FL.

Suppose A_{krum} is the Krum aggregation rule, and w_i is the local model that the ith worker device intends to send to the master device when there are no attacks. Without loss of generality, we assume the first c worker devices are compromised. Besides, w_i is the model before-attack, and w_i' is after-attack. Capital W refers to the global model. Thus, we have:

$$\text{Before attack: } W = A_{krum}(w_1, \dots w_c, w_{c+1}, \dots, w_m)$$

$$\text{After attack: } W_1' = A_{krum}(w_1', \dots w_c', w_{c+1}, \dots, w_m)$$

We denote by Γ_w^a the set of local models among the crafted c compromised local models and $m-c$ benign local models that are the closest to the local model w with respect to Euclidean distance. Moreover, we denote by $\tilde{\Gamma_w^a}$ the set of benign local models that are the closest to w with respect to Euclidean distance. If w_1' is chosen by Krum, we have the following:

$$\sum_{l \in \Gamma_{w_1'}^{m-c-2}} D(w_l, w_1') \leq min \sum_{l \in \Gamma_{w_i}^{m-c-2}} D(w_l, w_i) \tag{3.1}$$

Considering each malicious node could send the same w_1' to max the attack effect, we have

$$\sum_{l \in \Gamma_{w_1'}^{m-2c-2}} D(w_l, w_1') \leq min[\sum_{l \in \Gamma_{w_i}^{m-c-2}} D(w_l, w_i) + CD(w_1', w_i)] \tag{3.2}$$

Inequality (3.2) could approximately transfer to (3.3)

$$\sum_{l \in \Gamma_{w_1'}^{m-2c-2}} D(w_l, w_1') - min[\sum_{l \in \Gamma_{w_i}^{m-c-2}} D(w_l, w_i)] \leq CD(w_1', w_i) \tag{3.3}$$

In equation (3.3), as the subtrahend and c are fixed (i.e., the attackers cannot manipulate), the attack upper bound is only limited by $D(w_1', w_i)$ However, because any $D \in max\ C \sum_{l \in \Gamma_{w_i}^{m-c-2}} D(w_l, w_i)$ will be discarded; the upper bound should be:

$$max\ C \sum_{l \in \tilde{\Gamma}_{w_i}^{m-c-2}} D(w_l, w_i)$$

In other words, the attack surface is only organized by benign model updates and limits the max of the perturbation.

However, because of the non-IID character of federated learning, even benign gradients can still introduce a high Std and big attack surface. This brings difficulty to the existing FL methods for identifying, discarding malicious gradient, and reducing attack surface when the attackers stay close to the benign gradient boundary (i.e., G_0 and G_m in Figure 3.4). On

the contrary, in our Mini-FL, similar gradients are aggregated in the same group; the high inter-group similarity can lower Std and, hence, smaller attack surfaces. Suppose $G_0, G_1 \ldots G_m$ are m gradients uploaded from benign clients.

The following derivation demonstrates how our Mini-FL reduces the attack surface and defense against a state-of-the-art attack [37].

To achieve the attack goal and to avoid being discarded by the byzantine-robust aggregation rule, [37] sets the optimization problem as (3.4).

$$w_1' = w_{Re} - \lambda s \tag{3.4}$$

s is a column vector of the changing directions of all global model parameters (i.e., $s = 1 or -1$); w_{Re}presents the model received from sever this interaction and can be regarded as the initiation of crafting. Because of the training process, w_{Re} is unavoidable to be broadcast to all clients. Recall the Mini-FL method divides gradients received into different groups to generate the $w_g^{sub}, g = 1, 2, 3...$ (g indicates the different group), and subsequently calculates the weighted mean of $w_g^{sub}, g = 1, 2, 3...$ as w_{Re}. Thus, in Mini-FL, the attackers should use $w_g^{sub}, g = 1, 2, 3...$ to replace w_{Re} in each group g and transfers the optimization problem as (3.5) to attack different group g to achieve the same attack effectiveness in [37]:

$$w_g^{sub'} = w_g^{re} - \lambda s, g = 1, 2, 3... \tag{3.5}$$

However, because all group defining and gradients dividing work are only performed on the server side, all clients (including attackers) can only access w_{Re} instead of $w_g^{re}, g = 1, 2, 3...$, attackers cannot access the initiation of crafting in Mini-FL.

On the other hand, λ is the key to executing the attack [37]. In the proof proposed, λ is generated by solving the following inequality (3.6)

$$\sum_{l \in \Gamma_{w_1'}^{m-c-2}} D(w_l, w_1') \leq \min_{c+1 \leq i \leq m} \sum_{l \in \Gamma_{w_i}^{m-c-2}} D(w_l, w_i) \tag{3.6}$$

Then (3.6) transforms to (3.7) in [37]

$$\sum_{l \in \tilde{\Gamma}_{w_1'}^{m-2c-1}} D(w_l, w_1') \leq \min_{c+1 \leq i \leq m} \sum_{l \in \Gamma_{w_i}^{m-c-2}} D(w_l, w_i) \tag{3.7}$$

We can find that the only difference is the research replaces the range of l from (3.8) to (3.9)

$$l \in \Gamma_{w_1'}^{m-c-2} \tag{3.8}$$

$$l \in \tilde{\Gamma}_{w_1'}^{m-2c-1} \tag{3.9}$$

The necessary and sufficient condition of this replacement is the research assumes the other $c-1$ malicious models can stay close with w_1' even be the same (i.e., The distance between w_1' and the other $c-1$ compromised local models is 0). However, the malicious nodes are sent to different groups and cannot stay the same because of grouping. Hence the result of the (3.7) will be represented as (3.10) under the Mini-FL framework:

$$\lambda \leq \frac{1}{\sqrt{d}}[\min_{c+1 \leq i \leq m} D(w_l, w_i) + \max_{c+1 \leq i \leq m} D(w_l, w_{Re})] \tag{3.10}$$

Recall λ in research[37] has been limited as (3.11)

$$\lambda \leq \frac{1}{(m-2c-1)\sqrt{d}} \min_{c+1 \leq i \leq m} (\sum_{l \in \tilde{\Gamma}_{w_i}^{m-c-2}} D(w_l, w_i)) + \frac{1}{\sqrt{d}} \max_{c+1 \leq i \leq m} D(w_i, w_{Re}) \tag{3.11}$$

As the upper bound of (3.10) is smaller than (3.11), λ's upper bound has been reduced by Mini-FL (Mini-Krum).

Because the initiation of crafting w_{Re} can not be assessed by malicious nodes and the upper bound of λ has been reduced, Mini-FL effectively relieves the attack [37] Furthermore, the Mini-FL method collects the information in each heterogeneous clusters which guarantee the information of different classes can be evenly learned by the global model even in extreme non-IID settings.

We further provide a visual representation (Figure 3.4) to demonstrate the effectiveness of Information Asymmetry and Attack Surface Reduction. Under the existing FL methods,

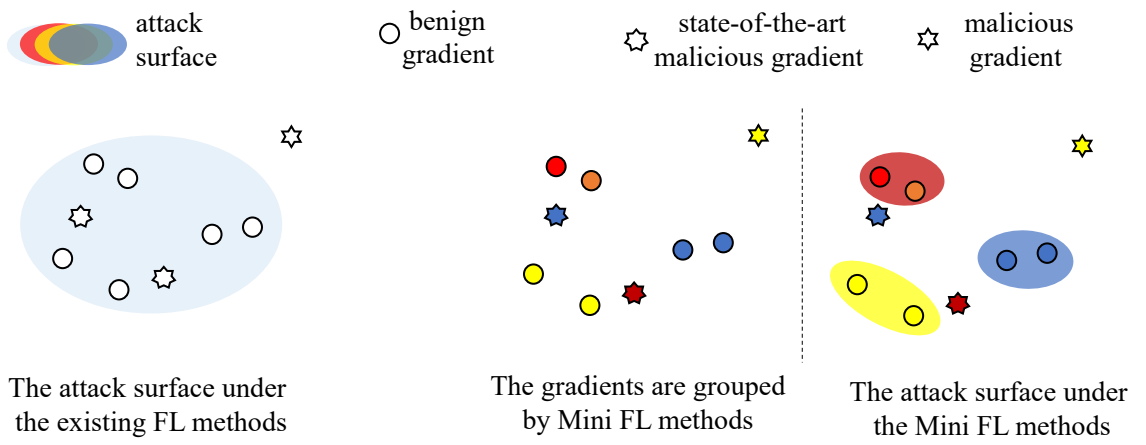

FIGURE 3.4: The effectiveness of information asymmetry and attack surface reduction.

primary malicious gradients can be effectively detected and discarded as it introduce a fixed, large perturbation and consequently acts as an outlier. However, state-of-the-art attackers can mislead the global aggregation when leveraging statistics and staying close to the boundary benign gradients. In our proposed Mini FL methods, gradients from clients are assigned to distinct groups based on a predefined grouping principle. Due to Information Asymmetry, attackers lack knowledge about the grouping principle and attempt to remain close to the boundary using statistical methods. During Mini-FL aggregation, each group conducts aggregation independently; malicious gradients near the boundary clusters but belong to other groups. As a result, attackers are considered outliers within each group and can be effectively identified. We can find the original whole attack surface has been reduced and replaced with several small attack surfaces.

3.6 Evaluation

3.6.1 Experimental Setup

A) Evaluation Dataset

We use different datasets to evaluate our Mini-FL framework, which includes MNIST [31] and Fashion-MNIST [152]. Different from the IID settings, under non-IID settings, the data

from each client are drawn from the different label distributions. In this paper, we consider the most common non-IID setting that the data volume of each client biases across different labels. Specifically, clients' data may be mainly occupied by different labels because of the interest heterogeneity.

To simulate the dataset pattern mentioned, we set different non-IID degrees when distributing training data. Suppose U is the universe of the data labels in the learning task; we set the training data size of each client as s and assign $p * s$ training examples with label L[1] ($L \in U$) to the client i to simulate the data of client i biases on the label L, where p is the probability. Then, we fill up the rest $s - (p * s)$ training data to this client evenly with other classes' data in $\complement_U L$. We note that the data of client i will evenly consist of all classes (not biased) if $p = \frac{1}{|U|}$, and only include class L if $p = 1.0$ As the possibility parameter p controls the distribution of training data on clients, we call p as the non-IID degree. To further embody the source of each non-IID distribution, we assign a feature (i.e., Geo, Time, or User feature) for each item of local model updates.

MNIST Variants: The MNIST [31] (Modified National Institute of Standards and Technology) database is an extensive database of handwritten digits that includes 60,000 training images and 10,000 testing images. To simulate people's different handwriting habits in different countries [65], we divide clients into five groups; each group owns one unique IP range (reflect different countries) and training examples with two different labels (reflect different handwriting habits). We use MNIST-1.0 ($p = 1.0$) to simulate the extreme non-IID situation (non-IID degree=1.0). In other words, each group only has two different unique labels of training examples in MNIST-1.0. Furthermore, we use MNIST-0.75 and MNIST-0.5 to evaluate the effectiveness of Mini-FL in different non-IID degrees. MNIST-0.75 and MNIST-0.5 have similar settings as MNIST-1.0, but the non-IID degree p is 0.75 and 0.5, respectively.

Fashion MNIST Variants: The FMNIST (Fashion MNIST) [152] dataset includes 60,000 gray-scale images of 10 fashion categories (T-shirt/Top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot) and a test set of 10,000 images. To simulate the changing

[1]The L could be a single element or a set; in other words, the data could bias on one label or several labels.

trend of the fashion dress, we divide clients into five groups; each group belongs to a time window (i.e., Years, months) and has training examples with two different labels (reflect the popular dress). Similar to MNIST-1.0, we set $p = 1.0$ for Fashion-MNIST-1.0 to simulate the extreme non-IID setting. Furthermore, we generate FMNIST-0.75 and FMNIST-0.5 to reflect the different non-IID degrees of the Fashion-MNIST dataset. We set $p = 0.75$ and $p = 0.5$ for Fashion-MNIST-0.75 and Fashion-MNIST-0.5, respectively.

B) Evaluated Poisoning Attacks

Mini-FL provides a new framework to enhance the security of FL, and the excellent generalization enables Mini-FL can introduce most existing byzantine-robust aggregation rules. We introduce Krum [9], Trimmed-mean [167], and Median [167] in experiments, respectively, and select the following poisoning attacks to evaluate the effectiveness of Mini-FL; we have not introduced FL-Trust in Mini-FL as FL-Trust does not fit extreme non-IID scenarios — Krum attacks can achieve 90% attack success rate when the root dataset's bias probability is over 0.6 [16]. We follow the threat model formed in Section 3.3 that assumes some clients have been controlled by the adversary and participant in the learning task each iteration. These malicious clients can craft the poisoned gradient based on the data owned by the original benign client through one of the following attack strategies. The data distribution and group information of the malicious client is shown in Table 3.7.

We follow the attack strategies introduced in Section 3.2 and select six different attacks in the experiments. The attack strategies selected include three primary attacks (i.e., "Reverse attack," "Random attack," "Partial Drop attack") and three state-of-the-art attacks (i.e., "Little is enough attack," "Fall of empires attack," "Local model poisoning attack"). The details of these attacks and settings are shown as follows:

I. Reverse Attack: Reverse Attack poisons the global model by uploading the reverse gradient. We follow the setting in [29] and set the attack multiple as 100.

II. Random Attack: Random Attack poisons the global model by uploading a random gradient; we set the random activation probability as 80%.

III. Partial Drop Attack: Partial Drop Attack masks the gradient parameter as 0 with probability p. As the parameter naturally carries a few 0 in our training tasks, we enhance the attack

strength by replacing the mask 0 as -1 and setting p as 0.8 in experiments.

IV: Little is Enough Attack: Little is Enough Attack leverages the dimension curse of machine learning and uploads the crafted gradient where gradient = μ + z * σ; here, μ and σ are the mean and standard deviation of the gradients respectively. z is the attack multiple, and we set z as 1.035 and 2.035.

V: Fall of Empires Attack: Fall of Empires Attack uploads the crafted gradient where gradient = $-z * \mu$. Here, μ is the mean of gradients and z is the attack multiple; we set z as 1 and 10.

VI: Local Model Poisoning Attack: Local Model Poisoning Attack infers the convergence direction of the gradients and uploads the scaled, reverse gradient to poison the global model. We follow the default setting in [37] for the local model poisoning attack.

C) Evaluation Metrics

Since these attacks (i.e., untargeted attacks) aim to reduce the model's global accuracy indiscriminately, we use the testing accuracy to evaluate the effectiveness of our Mini-FL. In particular, we use a part of the data owned as testing examples and test the model's global accuracy in each iteration. The testing accuracy reflects the model's robustness against byzantine attacks; in other words, it is more robust if the model has a higher testing accuracy. We further use the existing FL methods with the original framework as the baseline to compare against.

D) FL System Setting

Without other specific notifications, we use the following setting.

Global Model Setting: As this study does not aim to improve the model accuracy through crafting the model, we use a general model for training MNIST and Fashion-MNIST. This model consists of a dense layer (28 * 28) and a softmax layer (10).

Learning Parameters: We set the learning rate as 0.01, the batch size as 128, and the epoch as 50. We set the global iterations as 300 for MNIST and 500 for Fashion-MNIST. As some byzantine-robust methods (Krum in this study) require the parameter M for the upper bound of the number of malicious clients, we follow the setting in [9] that the server knows the exact number of all malicious clients. However, since Mini-FL defines groups and performs

aggregation accordingly, Mini-FL further requires the malicious clients m of each group when introducing Krum. To maintain the generality, We set m to belong to the group size:

$$m = \frac{n_{group}}{N_{global}} M$$

Here, n_{group} is the client number of the group (i.e., group size), and N_{global} is the total amount of clients. In other words, we do not give any privilege to Mini-FL, and Mini-FL can only use the proportion to infer the number of malicious clients in each group.

Clients & Data Distribution: We assume 20 clients participate in the learning task in each iteration, and 25% of clients are malicious. In Mini-FL, gradients are assigned in different groups as they carry different features. To simulate the non-IID setting in the real world, we assign different numbers of clients to different groups; subsequently, the larger group has more malicious clients. Table 3.7 illustrates the setting detail for the MNIST and Fashion-MNIST (non-IID degree=1.0).

TABLE 3.7: Illustration of the setting (Client & Data) for the MNIST and Fashion-MNIST (non-IID degree=1.0).

Groups	**Group1**	**Group2**	**Group3**	**Group4**	**Group5**
Training Labels	1, 2	3, 4	5, 6	7, 8	9, 0
Client ID	C1, C6 C11, C16, C19	C2, C7 C12, C17, C20	C3, C8 C13, C18	C4, C9 C14	C5, C10 C15
Attackers	C1,C11	C2,C12	C3	None	None

3.6.2 Experimental Results

The results show the existing FL methods can not effectively aggregate the information of different classes in non-IID scenarios; besides, the Mini-FL achieves better robustness than the existing FL methods. Figures 3.5, 3.6, and 3.7 illustrate the global accuracy of the existing FL methods/ Mini-FL methods under different non-IID degrees. When increasing the non-IID

degree, the results show that most Mini-FL methods can maintain a similar global accuracy under the same attack, while the existing FL methods witness decreasing global accuracy. For instance, Mini-median stably maintains around 90% global accuracy against various attacks and non-IID settings. In contrast, Median achieves around 85% global accuracy against various attacks in MNIST-0.5 but drops global accuracy to 64.62%, 26.51%, 55.79%, and 53.68% in MNIST-1.0 under Little attack, Fall attack, Random attack, and Partial Drop attack, respectively.

Mini-FL achieves the defense objectives. Recall that the defense objectives include two parts (see Section 3.3.2): Achieving byzantine-robustness against untargeted attacks and Maintaining the data minimization principle of FL. The experimental results show our Mini-FL framework achieves these goals.

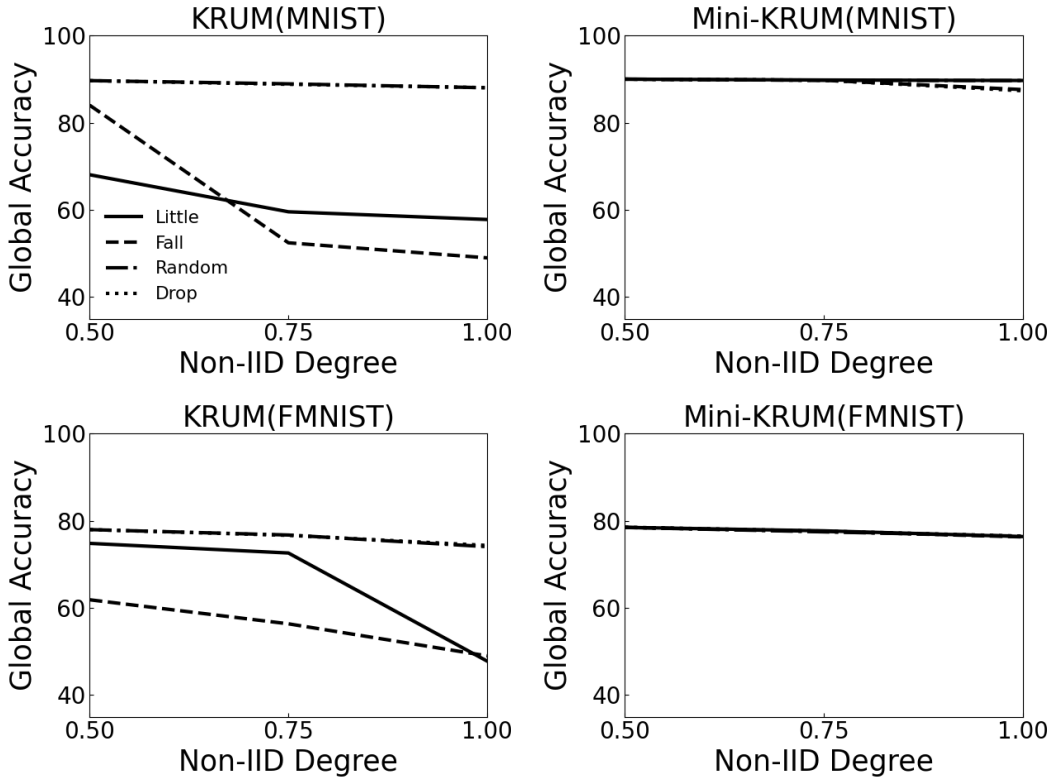

FIGURE 3.5: The robustness comparison between Krum and Mini-Krum methods under different non-IID degrees.

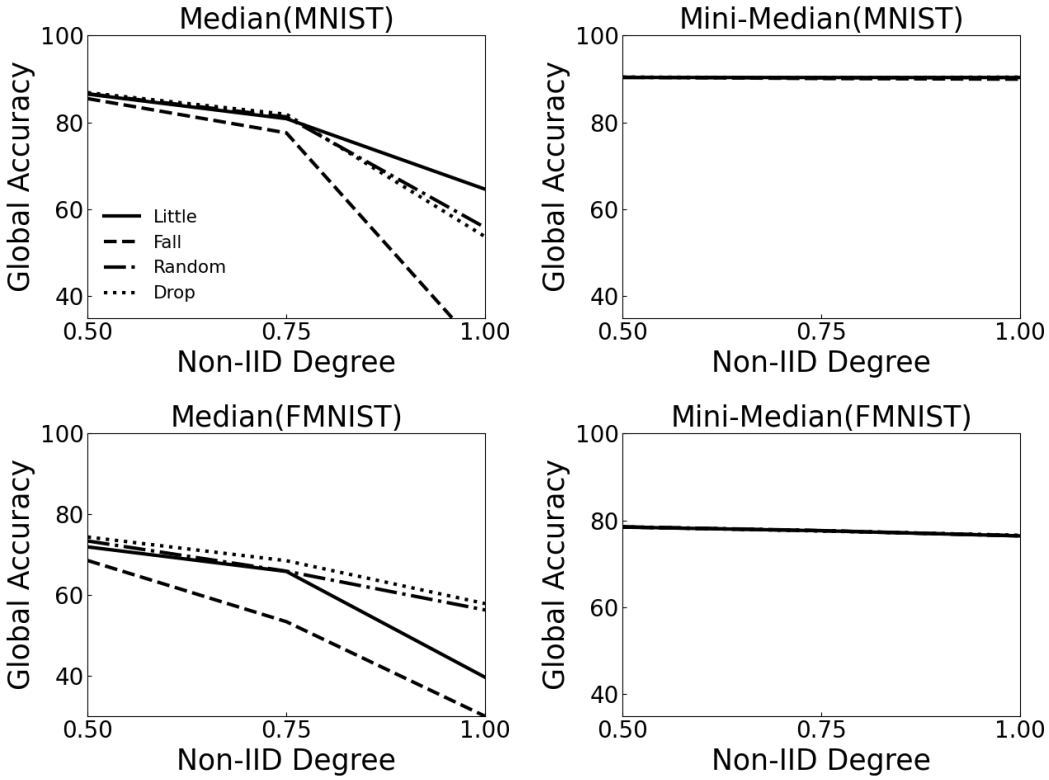

FIGURE 3.6: The robustness comparison between Median and Mini-Median methods under different non-IID degrees..

First, Mini-FL achieves similar global accuracy as FedAvg (average aggregation rule, also called Vanilla) in the non-attack setting, but most existing byzantine robust FL methods have a decreased accuracy. We consider the reason includes the existing FL methods can not effectively aggregate the information of different clients(with corresponding classes) and defend against the adversary in non-IID settings. For instance, FedAvg and all Mini-FL methods (i.e., Mini-Krum, Mini-Median, Mini-Trimmed mean) achieve over 90% global accuracy on MNIST-0.75 while Krum, Median, and Trimmed mean get 77.27%, 80.45%, 88.29%, respectively; on FMNIST-1.0 (i.e., Fashion MNIST-1.0), FedAvg and all Mini-FL methods achieve the global accuracy around 77%, while 61.25%, 43.88%, and 72.89% for Krum, Median, and Trimmed mean, respectively. Table 3.8 illustrates the global accuracy of different FL and Mini-FL methods under different non-IID degrees and non-attack settings. The result shows the Mini-FL framework increases the accuracy of existing FL methods in the non-attack scenario. This may be because benign gradients could be very different in the

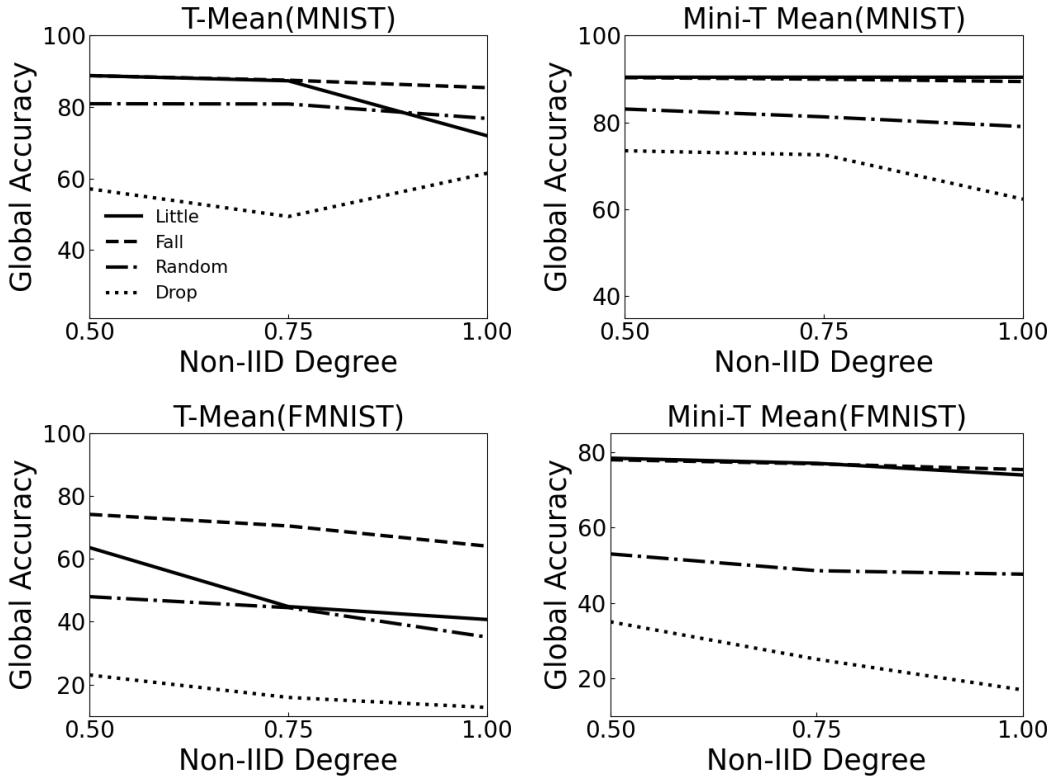

FIGURE 3.7: The robustness comparison between Trimmed-Mean and Mini-Trimmed Mean methods under different non-IID degrees..

non-IID setting, which may be regarded as malicious gradients and discarded by the existing FL method. As Mini-FL performs the aggregation by groups, it could comprehensively collect features from different groups and guarantee global accuracy.

Second, our Mini-FL shows better robustness and stability than most existing FL methods against different attacks and under different non-IID settings. Specifically, most Mini-FL methods can maintain unattacked global accuracy even when facing a state-of-art attack and under an extreme non-IID setting; on the contrary, existing FL methods immensely decrease global accuracy and even be fully broken. For instance, the Mini-median achieves 89.77% global accuracy in MNIST-1.0 under Local Poison attack, while the Median method drops global accuracy from 53.60% to 2.85%. Table 3.9 illustrates the global accuracy of the existing FL and Mini-FL methods under different non-IID degrees and different attacks.

TABLE 3.8: The global accuracy of the existing FL and Mini-FL methods under different non-IID degrees and non-attack settings.

Dataset	**MNIST-1.0**	**MNIST-0.75**	**MNIST-0.5**
Average	88.89%	90.04%	90.38%
Krum	88.25%	77.27%	87.47%
Mini Krum	89.51%	90.03%	90.09%
Median	53.60%	80.45%	86.30%
Mini Median	90.34%	90.26%	90.47%
Trimmed-Mean	86.88%	88.29%	89.24%
Mini Tri-Mean	90.38%	90.47%	90.51%
Dataset	**Fashion MNIST-1.0**	**Fashion MNIST-0.75**	**Fashion MNIST-0.5**
Average	79.39%	77.33%	78.48%
Krum	61.25%	65.80%	75.39%
Mini Krum	76.51%	77.51%	78.49%
Median	43.88%	65.87%	71.19%
Mini Median	76.54%	77.74%	78.56%
Trimmed-Mean	72.89%	74.92%	77.56%
Mini Tri-Mean	76.53%	77.50%	78.47%

Moreover, the result shows that although the Mini-trimmed mean improves the robustness of the trimmed mean method, it achieves lower global accuracy than other Mini-Fl methods. For instance, the Mini-trimmed mean achieves 62.33% and 16.88% global accuracy under drop attack in MNIST1.0 and Fashion MNIST-1.0 while other Mini-FL methods get around 90% and 76.5%, respectively. This is because the original FL method (Trimmed mean(β=20%)) draws a larger attack surface than Krum and Median as Trimmed mean(β=20%) accept and aggregates 80% gradients received while Krum and Median accept only one gradient.

Third, Mini-FL maintains the principles of focused collection and data minimization of FL. All of the information used for grouping (i.e., IP address, response time, and client ID) is naturally carried by the gradients when uploading. Mini-FL neither asks clients to upload

TABLE 3.9: The global accuracy of the existing FL and proposed Mini-FL methods when defending against various model poisoning attacks under different non-IID degrees.

(a) MNIST-1.0							
	Average	Krum[9]	Mini Krum	Median[167]	Mini Median	T-Mean[167]	Mini T-Mean
Little[6], z=2.035	74.71%	74.92%	89.62%	53.76%	90.37%	52.83%	89.76%
Little[6], z=1.035	84.42%	57.78%	89.70%	64.62%	90.34%	71.95%	90.40%
Fall[154], eps=10	23.73%	77.34%	88.38%	54.98%	90.10%	61.54%	90.18%
Fall[154], eps=1	78.23%	48.97%	87.61%	26.51%	89.95%	85.41%	89.41%
Random[99]	80.19%	88.03%	89.68%	55.79%	90.37%	76.80%	79.04%
Partial Drop[99]	61.65%	88.05%	87.33%	53.68%	90.42%	61.47%	62.33%
Local Poison[37]	78.62%	N/A	N/A	2.85%	89.77%	64.51%	87.32%
(b) FMNIST-1.0							
	Average	Krum[9]	Mini Krum	Median[167]	Mini Median	T-Mean[167]	Mini T-Mean
Little[6], z=2.035	64.75%	74.15%	76.35%	43.79%	76.76%	29.66%	58.82%
Little[6], z=1.035	70.51%	47.76%	76.36%	39.56%	76.43%	40.72%	73.98%
Fall[154], eps=10	35.57%	74.35%	76.22%	44.18%	76.21%	38.48%	76.60%
Fall[154], eps=1	48.67%	48.97%	76.28%	26.91%	76.44%	64.10%	75.43%
Random[99]	66.95%	74.04%	76.44%	56.26%	76.47%	35.12%	47.64%
Partial Drop[99]	19.78%	74.33%	76.46%	57.84%	76.58%	12.69%	16.88%
Local Poison[37]	45.88%	N/A	N/A	2.82%	76.26%	45.97%	60.51%
(c) MNIST-0.75							
	Average	Krum[9]	Mini Krum	Median[167]	Mini Median	T-Mean[167]	Mini T-Mean
Little[6], z=2.035	66.89%	83.84%	89.74%	81.25%	90.22%	61.05%	89.94%
Little[6], z=1.035	89.49%	59.55%	89.81%	80.65%	90.34%	87.33%	90.41%
Fall[154], eps=10	85.33%	77.27%	89.77%	79.15%	89.98%	64.09%	90.23%
Fall[154], eps=1	89.63%	52.43%	89.74%	77.59%	90.10%	87.51%	89.95%
Random[99]	74.85%	88.93%	89.74%	81.28%	90.24%	80.85%	81.28%
Partial Drop[99]	69.99%	88.83%	89.77%	81.87%	90.31%	49.36%	72.53%
Local Poison[37]	85.16%	N/A	N/A	62.31%	90.00%	75.90%	88.78%
(d) FMNIST-0.75							
	Average	Krum[9]	Mini Krum	Median[167]	Mini Median	T-Mean[167]	Mini T-Mean
Little[6], z=2.035	31.07%	76.48%	77.52%	65.26%	77.84%	36.18%	71.30%
Little[6], z=1.035	61.64%	72.59%	77.68%	65.81%	77.62%	44.84%	77.10%
Fall[154], eps=10	64.79%	76.32%	77.64%	64.11%	77.69%	42.56%	77.71%
Fall[154], eps=10	75.72%	56.33%	77.55%	53.35%	77.63%	70.54%	76.95%
Random[99]	28.77%	76.72%	77.44%	65.87%	77.66%	44.50%	48.56%
Partial Drop[99]	22.18%	76.65%	77.62%	68.47%	77.56%	15.88%	25.05%
Local Poison[37]	61.03%	N/A	N/A	31.84%	77.51%	51.57%	63.58%
(e) MNIST-0.5							
	Average	Krum[9]	Mini Krum	Median[167]	Mini Median	T-Mean[167]	Mini T-Mean
Little[6], z=2.035	79.15%	88.71%	90.02%	86.56%	90.36%	80.83%	90.34%
Little[6], z=1.035	89.88%	68.06%	90.01%	86.51%	90.35%	88.80%	90.41%
Fall[154], eps=10	88.38%	89.66%	90.03%	85.88%	90.38%	71.64%	90.35%
Fall[154], eps=1	90.11%	84.03%	89.99%	85.48%	90.41%	88.77%	90.30%
Random[99]	78.43%	89.67%	90.02%	86.60%	90.36%	80.92%	83.08%
Partial Drop[99]	72.06%	89.66%	90.00%	86.86%	90.43%	57.11%	73.48%
Local Poison[37]	86.37%	N/A	N/A	80.68%	90.28%	76.48%	89.81%
(f) FMNIST-0.5							
	Average	Krum[9]	Mini Krum	Median[167]	Mini Median	T-Mean[167]	Mini T-Mean
Little[6], z=2.035	37.92%	77.80%	78.48%	73.12%	78.57%	42.14%	74.07%
Little[6], z=1.035	73.56%	74.80%	78.48%	71.89%	78.46%	63.63%	78.46%
Fall[154], eps=10	70.55%	77.97%	78.46%	71.56%	78.51%	52.36%	78.47%
Fall[154], eps=1	77.17%	61.85%	78.45%	68.52%	78.47%	74.23%	78.04%
Random[99]	33.06%	77.96%	78.47%	73.36%	78.54%	48.00%	53.02%
Partial Drop[99]	29.65%	77.96%	78.52%	74.35%	78.51%	23.08%	34.90%
Local Poison[37]	29.65%	N/A	N/A	63.93%	78.33%	70.18%	76.01%

TABLE 3.10: Illustration of the data processing in the case study.

Range ID	Whole Range	→	Range1	Range2	...	Range9	Range10
DIS Range	3.5 ∼ -1.5	→	3.5 ∼ 3.0	3.0 ∼ 2.5	...	-0.5 ∼ -1	-1 ∼ -1.5

their information further nor digs their features through reverse engineering, which provides the same privacy protection as the existing FL methods.

3.6.3 Case Study

In this section, we provide a case study of the Boston House Price Forecast to further demonstrate the Mini-FL work process and evaluate the practicality and effectiveness of Mini-FL in the real world.

A) Case Study Setup

Dataset and Data Deviation: Boston house price dataset [48] records 13 features (e.g., crime rate, pupil-teacher ratio, etc.) of 506 sample houses in Boston. Although Boston house price [48] has not directly recorded the address of each property, the feature DIS (i.e., weighted distances to Boston employment centers) could be alternatively regarded as the address of each house. To simulate the federated learning scenario that different clients participating in the learning task at different addresses, we equally divide the whole DIS range into ten sub-ranges; according to the DIS sub-range, all data are assigned into these ten groups subsequently, Table 3.10 illustrates the data processing.

Model Architecture and FL System Settings: We train a deep neural network to predict the Boston house price; this model consists of three dense layers with two Relu activations: Dense+Relu (13*32), Dense+Relu (32*16), Dense (16*1).

We set 0.01, 10, and 10 as the learning rate, batch size, and epoch, respectively, and train the global model with 300 iterations. We assume that 12 clients (from different addresses) participate in the learning task, and 25% are malicious.

Adversary, The Existing FL, Mini-FL Methods, and Evaluation Metric: We use Median [167] and Mini-Median as the control group and select Reverse attack [8] ($t = 10$) and a state-of-the-art attack (Fall of empires attack) [102] ($eps = 10$) to evaluate the robustness of each method. We use the loss value of the global model as the evaluation metric; specifically, the FL method is more robust if its global model achieves a lower loss under attacks.

B) Mini-FL

We follow the Mini-FL proposed in Section 3.4.2. As only the Geo-feature (DIS) is available in this case, we set Geo-feature (DIS) as the grouping feature and perform the "elbow method" for all gradients received to detect the cluster. The 'elbow method' shows the gradients received can be divided into 6 clusters; Figure 3.8 illustrates the "elbow method" outcome. Based on the result, we further divide the clients into 6 groups and generate the grouping policy. Table 3.11 illustrates the grouping policy.

In further iterations, we follow the grouping policy to distribute the gradients received in different groups and perform the Median method to generate the grouped model respectively. Then, we utilize the weighted mean proposed in Section 3.4.2 to update the global model and finish each learning iteration.

TABLE 3.11: The grouping policy of Boston House Price Forecast learning task.

Group ID	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Client ID	C1, C2, C3	C4	C5	C6	C7, C8, C9, C10, C11	C12

C) Case Study Experimental Results

The experimental results show our Mini-FL method is more robust and achieves a lower loss. Specifically, the Median achieves 31 and 47 global loss values under the reverse and Fall attacks, but the Mini-Median decreases the global loss to 26 and 24, respectively. Figure 3.9 illustrates the global loss value of the existing Median method and proposed Mini-Median method when defending against Reverse and Fall attacks.

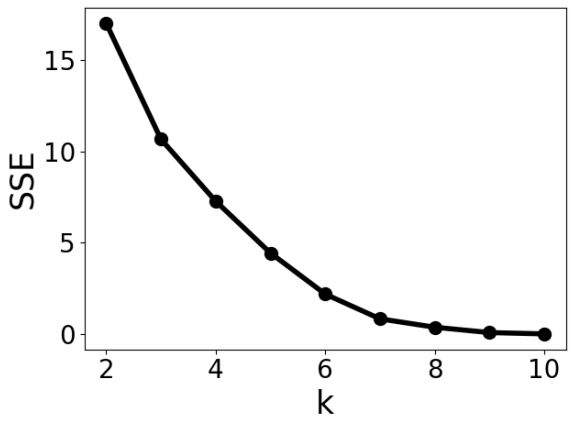

FIGURE 3.8: Illustration of the elbow method outcome in Boston House Price Forecast learning task.

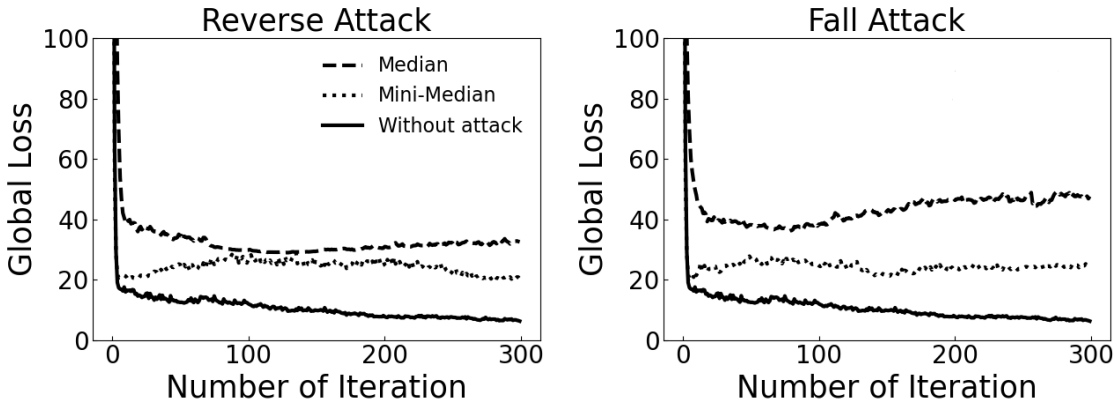

FIGURE 3.9: The robustness comparison between Median and Mini-Median.

3.7 Discussion

A) Mini-FL and CFL

We note that the CFL [123] algorithm has been designed based on a strong assumption that "only the largest cluster is benign and all other clusters are adversarial," which is valid only under the perfect IID settings. Once the benign clients behave heterogeneously, the expected large cluster will separate into several sub-clusters, putting CFL in the dilemma of selecting. Instead of identifying and separating the adversary, our Mini-FL introduces clustering methods to reduce the attack surface. Furthermore, Mini-FL considers a more practical non-IID

scenario; we demonstrate that Mini-FL can effectively enhance the robustness of FL even though several (more than one) benign clusters exist.

B) Mini-Krum and Bulyan

Mini-Krum and Bulyan [45] are different, although both of them rely on performing Krum and trimmed-mean methods. Specifically, Mini-Krum performs Krum by group and generates the weighted average as the global model. In contrast, Bulyan[45] globally performs Krum n times to select n gradients and performs Trimmed-mean to generate the global model. As Bulyan[45] does not consider the non-IID setting of FL, it faces a similar degraded performance as other FL methods in non-IID scenarios.

C) non-IID Sources

In this research, we select Geo-feature, Time-feature, and User-feature as the grouping feature candidates as they are the most common source of non-IID in the real world and available to be collected in most scenarios without breaking participant's privacy. Except for the scenarios aforementioned in Section I, the Geo-feature derived scenarios may further include natural disaster exploration [5], customer taste analyzing[95], etc. The Time-feature derived scenarios may include: pandemic pattern evaluation [62, 1], consumer behavior preferences[118, 113], road monitoring[46], etc. The User-feature derived scenarios may include (network) user profile generation[77, 114], etc. However, we note that the non-IID source could be more complicated in some particular settings and even be a combination in some cases[65, 180, 181]. We leave investigating further to explore other non-IID sources and combination possibilities and further improve the Mini-FL method.

D) Evaluation Dataset

The effectiveness of our proposed Mini-FL framework is dependent on the grouping principle used to classify the received gradients, i.e., the definitions of non-IID features and their boundaries, rather than the gradients themselves. To emulate the real-world Federated Learning process, we assign each gradient in our experiments with identified non-IID features, namely IP address, Time Window, and User ID. We note that employing the same distribution strategy for non-IID features will yield similar experimental results across different evaluation datasets.

3.8 Conclusion

In this Section, we evaluate the robustness of existing FL methods in different non-IID settings and subsequently propose a new framework called Mini-FL to enhance Federated Learning robustness. The main difference between Mini-FL and existing FL methods is that Mini-FL considers FL's non-IID nature and performs the byzantine tolerant aggregation in different groups. Our evaluation shows that Mini-FL effectively enhances existing FL methods' robustness and maintains a stable performance against untargeted model attacks and different non-IID settings.

CHAPTER 4

Defending Against FL Poisoning Attack

In this chapter, we present our solution to defend against the FL poisoning attacks. We investigate two representative FL poisoning attacks, namely Label Flipping Attack and Class Imbalance Attack, and then analyze how to defend these, respectively. To defend Label Flipping Attack, we propose Honest Score Client Selection based FL (HSCSFL), which relies on a novel client selection scheme, namely Honest Score Client Selection (HSCS), to monitor the global model, prioritize the ranking, and select the most truthful clients for participation in the aggregation process. To defend the Class Imbalance Attack, we propose Class Balanced FL to dynamically distribute weight in aggregation and guarantee the global model can achieve good learning performance across different classes. Two proposed algorithms are evaluated through different datasets and settings to demonstrate their robustness. Our HSCSFL algorithm is currently under review at Information Fusion under the name Honest Score Client Selection Scheme: Preventing Federated Learning Label Flipping Attacks in non-IID Scenarios. Our Class-Balanced FL algorithm is currently under review at Information Sciences under the name Contribution-Wise Byzantine-Robust Aggregation for Class-Balanced Federated Learning.

4.1 Preventing Label Flipping Attack

4.1.1 Introduction

Federated Learning (FL) [97] is a distributed machine learning (ML) paradigm that enables many clients to jointly train a model under a service provider (i.e., server) without sharing

the raw data. Although the Federated Learning system maintains the participants' privacy by enabling clients to locally train their model, it introduces a new attack surface for the adversary and makes FL vulnerable to poisoning attacks [176, 43, 103, 65]. In particular, the adversary can poison the global model and degrade its performance by uploading crafted gradients. Examples of poisoning attacks include "model poisoning attack" [17, 37, 6, 154] and "data poisoning attack" [78, 166, 136] which directly manipulates the report and crafts the training data to generate the poisoning gradient, respectively. As the most representative data poisoning attack, the Label Flipping Attack has been introduced by [78, 136]. In the Label Flipping Attack, the adversary first agrees on the attack strategy, which includes the attack class(es), target class, attack timing, etc. Then, the adversary flips the attack class of training data to the target class and uses the crafted training data to train the model; the poisoned gradient is consequently generated and submitted to the server.

To defend against the poisoning attacks and enhance the robustness of the FL system, byzantine resilient federated learning has been introduced by recent researchers [88, 16, 167, 9]. By leveraging the evaluation of the statistics, these works decrease the weight of suspicious clients in aggregation or select the most honest client as the aggregation outcome. For instance, Krum [9] calculates the sum Euclidean distance for a client's gradients received and selects the gradient achieving the shortest distance as the aggregation outcome in the current iteration. However, we note that most of these FL methods are designed for model poisoning attacks. As data poisoning attacks introduce smaller perturbations, the performance when defending against data poisoning attacks, especially Label Flipping Attack [78, 136], still remains unclear. Furthermore, the learning scenario of real-world FL applications is usually Non-Independent and Identically Distributed (non-IID) [65, 88]. The clients' training data and corresponding training models are usually heterogeneous due to the different user habits, locations, etc., which also brings difficulty for the existing FL methods to defend against poisoning attacks.

In this section, we first evaluate the robustness of the existing byzantine resilient FL methods [16, 167, 9] when defending against Label Flipping Attack [78, 136] in both IID and non-IID settings. The experimental results show that the existing FL methods can maintain robustness

in IID scenarios but fail to defend against such attacks [78, 136] under non-IID settings. To mitigate the research gap, we propose the Honest Score Client Selection scheme (short as HSCS) and the corresponding Honest Score Client Selection FL (short as HSCSFL). In HSCSFL, the server collects a clean evaluation dataset that evenly includes data for each class. During each learning iteration, the server first evaluates and records each class's performance of the global model; the class achieving low accuracy is consequently assigned a high-risk value. We call the recording of global model performance and the corresponding risk values as performance vectors ($PerV$) and risk vectors ($RisV$), respectively. Then, HSCSFL evaluates the class performance of each client's model and generates the accuracy vector ($AccV$) for each client. Finally, the server calculates the dot product of the risk vector and the accuracy vector for each client as its honest score (HS); only the clients with the top $p\%$ honest score are included in the following averaging aggregation. We evaluate our HSCSFL in MNIST and FMNIST with different non-IID degrees and different label-flipping strategies; the experimental result shows our HSCSFL can maintain robustness when defending against Label Flipping Attacks. [78, 136]

4.1.2 Related Work

Federated Learning Applications

As a collaborative training framework for machine learning that safeguards privacy, FL has garnered widespread attention from the industry. Today, numerous real-world FL applications have been proposed and deployed. These applications can generally be divided into three categories, including applications for smart devices [161, 130, 59], industrial engineering [53, 47] and healthcare domains [71, 56].

First, with Google's successful deployment of the user input prediction keyboard (Gboard) based on federated learning, applications of federated learning in smart devices have become a hot exploration concept.This category of applications includes not only text prediction of user input [161] but also the prediction of emojis [59] and the behavior of device users [130]. Second, given the achievement of FL in privacy preservation, an increasing number of FL based industrial engineering applications are being developed and deployed, especially in

data-sensitive fields within industrial engineering. For instance, the study [53] proposes a novel framework for environmental monitoring based on Federated Region Learning (FRL), designed to address the challenges associated with the exchange of monitoring data. Similarly, study [47] applies FL in industrial manufacture visual inspection task to detect defects in production tasks and guarantee the privacy for manufacturers. Third, FL has also been introduced in healthcare domain [71, 56] to to break down the barriers of analysis throughout different hospitals and provide disease prediction for patients. For example, the study [71] employs tensor factorization models within a federated learning framework for phenotyping analysis, extracting valuable insights from health records while preserving patient confidentiality by avoiding the sharing of patient-level data.

Poisoning Attacks on Federated Learning

Due to the distributed nature, Federated Learning is vulnerable to various poisoning attacks by malicious clients and untrusted participants [65, 103]. These poisoning attacks include model poisoning attacks and data poisoning attacks, which seek to damage the global model by directly crafting the gradients or local training data, respectively. Instead of directly adding perturbation on the gradient, the data poisoning attack [177, 78, 136, 42] generates the crafted gradient through poisoning training data. On the one hand, some adversaries [177, 42] aims to use crafted gradients to implant so-called "backdoor" into the global model. As a result, the learned model's outputs will be fixed to a preset target when facing specific inputs. These implanted backdoors could be semantics or graphics; for instance, [177] forces the model to generate the response "RACE people are psycho" when the user type "RACE people are," [42] can implant an imperceptible backdoor on the GoogLeNet [133], which leads the model to classify a panda as a gibbon when encountering the trigger. On the other hand, the adversary may seek to degrade the global model performance only on the target class by flipping the training labels [78, 136], which is the so-called "Label Flipping Attack." When performing such attack, the adversary decides the attack strategy which includes the attacked class, the destination class, the amount of attacked classes, and the attacking timing. Then, the adversary flips the training label following the attack strategy and uses the poisoning data to train the local model. Once the poisoned gradient continuously participates in the aggregation, the global model will misbehave on the attacked class. As "Label Flipping

Attack" does not directly manipulate the gradient, it introduces a minor perturbation, which makes it challenging to identify and defend.

Today, "label flipping attack" has been introduced in various real world applications and shows significant effectiveness. Study [134] applies this attack within a malware detection task for Android platforms, introducing a Silhouette Clustering-based Label Flipping Attack (SCLFA). This approach involves attackers calculating the silhouette value of each data sample and selecting those with a negative value as candidates to generate polluted data through label flipping. The paper [174] demonstrates that the "label flipping attack" also shows the effectiveness for fooling spam filtering systems. Through performing entropy method based flipping attacks, study [174] increases the false negative rate of Naive Bayes under the influence of label noise without affecting normal mail classification. On the other hand, the paper [125] introduces "label flipping attacks" in hardware Trojan detection systems. It demonstrates that model performance can be effectively deteriorated through a proposed stochastic hill-climbing search-based flipping attack, with only a small cost associated with flipping the labels.

Byzantine Resilient Federated Learning

To defend against various poisoning attacks and enhance the robustness of federated learning, Byzantine resilient FL methods have been proposed by recent researchers [88, 16, 167, 9]. Some of these works consider excluding the suspicious gradients from the aggregation base on the statistics. In particular, Krum [9] calculates the sum Euclidean distance (L2 distance) for a preset number of received gradients. The client that achieves the shortest distance is regarded as the most benign client, its gradient will become the new global gradient in this iteration. Trimmed Mean [167] and Median [167] are coordinate-wise aggregation methods that aggregate each parameter of the model, respectively. For each parameter, the server firstly sorts all the parameters received from client model updates. Then, the server trims the largest and smallest $p\%$ parameters and averages/takes the median value of the remaining gradients as the corresponding parameter of the new global model. FL Trust [16] enhances the FL robustness from the gradients' directions and magnitudes perspectives. Under the FL Trust framework, the server should collect a small clean dataset and keeps a corresponding

model. In each learning round, FL Trust calculates the cosine similarity between the owned and clients' gradients and assigns a higher honest score for those clients achieving high cosine similarity. The clients' gradients are normalized by the server gradient and consequently participate in the aggregation; the client with a higher honest score will receive a heavier weight and vice versa.

Given the substantial security challenges posed by "label flipping attacks", we focus on the these attacks [78, 136] in this work.We note that most of these byzantine resilient federated learning methods have been introduced to mitigate model poisoning attacks in IID settings; the effectiveness when defending against data poisoning, especially in non-IID scenarios still remains unclear.

4.1.3 Motivation

4.1.3.1 Background and Definition

Our problem is formulated on the synchronous federated learning (binary or multi-class) classifier task with a deep neural network (DNN) model. The DNN model consists of multiple layers, which are comprised of a corresponding set of nodes. During the learning process, each layer processes the input and sends the output to the following layers, where the first layer receives the training data as input, and the final layer generates the prediction result. FL paradigm enables multi-participants[1] collaboratively train a model on their local devices without sharing their raw data. Specifically, in each FL learning iteration, the server first broadcasts the current global model M to n clients. Then, each client uses the data owned to locally train the model M and sends the gradient g back to the server. Finally, the server aggregates all the model updates received by an aggregation method $\mathcal{A}(\cdot)$. The learning task starts from the server initializing the model and ends up with the global model M achieving a preset accuracy.

We use a joint distribution $p_i(x, y)$ to denote the private data owned by the ith participant, where $x = (x_1, ..., x_s)$ and $y = (y_1, ..., y_s)$ denote the feature space and the label space

[1] In this work, we combined use "participants" and "clients" with the same meaning.

respectively; the label $y_s \in C$, where C is the set of all possible classes. At the end of each learning iteration, the optimal model m_i^* of client n is the solution for the following optimization problem:

$$m_i^* = argmin_{m_i} F(m_i) \tag{4.1}$$

Here, $F(\cdot)$ is the expectation function:

$$F(m_i) = \mathbb{E}_{(x,y)\sim p_i}[l((x, m_i), y)] \tag{4.2}$$

where $l((x, m_i), y)$ represents the empirical loss. The joint distribution $p_i(x, y)$ could be composed as the following equation.

$$p_i(x, y) = p_i(x|y)p_i(y) \tag{4.3}$$

Here, $p_i(x|y)$ denotes the conditional distribution of feature x given label y, and $p_i(y)$ represents the marginal distribution of label y on the participant i. We note that in the real-world FL scenario, both the conditional distribution $p(x|y)$ and the marginal distribution $p(y)$ can be different across the clients due to its non-IID nature. On the one hand, different clients may have various representations under the same label (i.e., $p(x|y)$). On the other hand, clients may own different amounts of data for each class, which draws the heterogeneity of the marginal distribution (i.e., $p(y)$).

The differences between the local and optimal model of each client are subsequently calculated as the gradient g and sent back to the server. The server aggregates the gradients through the aggregation rule $\mathcal{A}(\cdot)$ once the preset amount of g has been received and subsequently uses it to update the global model,

$$\begin{aligned} g_i &= m_i^* - m_i \\ M^{t+1} &= M^t + \eta \cdot \mathcal{A}(g_i^t),\ i = 1, 2, ..., n \end{aligned} \tag{4.4}$$

where t denotes the iteration and η represents the learning rate which is a hyper-parameter.

During the learning process, clients may craft and upload a malicious g to poison the global model once penetrated by attackers. The attacker can directly manipulate the gradient g or poison the local training data by mismatching the feature x and the ground truth y pairs to generate the crafted gradient. To defend against such attacks, some byzantine-robust aggregation rules ($\mathcal{A}(\cdot)$) have been introduced. Instead of averaging the gradients received, these aggregation rules assign heavy weights to those gradients who behave honestly and vice versa.

In this paper, we focus on the label-flipping attacks (i.e., generate g by mismatching x and y) with different mismatching strategies. We consider that while the total volume of benign training data across each class is similar, the differences in marginal distributions primarily drive the non-IID nature in FL.

4.1.3.2 Effectiveness of the Existing Byzantine-Robust Aggregation Rules

In this section, we evaluate the effectiveness of the existing byzantine-robust aggregation rules when defending against the Label Flipping Attacks through MNIST in both IID and non-IID settings.

Experimental Setup

Threat Model: We assume 20 clients collaborate in the learning task to train a joint model M and 25% participants are controlled by the adversary, which aims to decrease the testing accuracy of the target class(es) y on the global model M through uploading crafted gradients. We consider the adversary can access the dataset on the controlled device belonging to the original client and utilize the dataset to generate the crafted gradient through the Label Flipping Attack strategy. Under the proposed data distribution and learning scenario, class 5 of MNIST receives the lowest testing accuracy in the non-adversarial setting (around 60%, shown in the next section) and has been widely recognized as class 8 by the model; we regard it as the most vulnerable class and set the attack strategy as "Flipping the training data label 5 as 8 on the attackers' device" to generate the crafted gradients in the proposed example.

Training Data: The MNIST database (Modified National Institute of Standards and Technology database) [31] is an extensive handwritten digits collection that includes a training set of 60,000 examples and a test set of 10,000 examples; each example has consisted of 28×28 pixels and a label of $0 \sim 9$ to indicate the class belonging to. Under the IID setting, we consider each client owes all training classes and evenly distribute the training data to all clients; we use $MNIST_{IID}$ to denote the case that MNIST training data with the proposed IID setting. In the non-IID setting, we consider the most general scenario that clients' data may have different marginal distributions ($p(y)$) because of the interest heterogeneity and leverage the parameter p (±0.025) to denote the non-IID degree. During data distribution, we assign the i client $p * s$ volume training examples with label y[2], where s is the preset training data volume of each participant. Then, we evenly assign other class training data to client i to fill up the rest $s - (p * s)$ capacity. We note that the client's data will be heavily biased on the label y if $p > \frac{1}{|U|}$, and only include the class y once $p = 1.0$, where U is the labels universe of the learning task when y is a single label. In the proposed example, we set $P = 0.9$ and use $MNIST_{0.9}$ to represent such a case.

Figure 4.1 illustrates an example of the data distribution for 20 clients in IID and non-IID (p=0.9) settings. Under the IID setting, all clients are evenly assigned each class data (left bar); under the non-IID setting, the clients' data bias on different classes, making up four different distributions (right four bars). We assume $Client_1 \sim Client_5$ play the attacker role.

Training Model: In this study, we use a general model for training, which includes a 28×28 dense layer and a $10\times$ softmax layer. We set the learning rate as 0.01 and batch size as 128 to train the model for 100 iterations.

Benchmarks and Metrics: As the Label Flipping Attack [78, 136] aims to reduce the accuracy of the targeted class, we use the accuracy as the metric, including global accuracy and the accuracy of each class. We select several representative byzantine robust aggregation FL methods (Krum [9], Median [167], Trimmed Mean [167] and FL Trust [16]) and evaluate their performance when facing Label Flipping Attack. We further evaluate the learning

[2]The y could be a single element or a set; in other words, the data could be biased on one label or several labels.

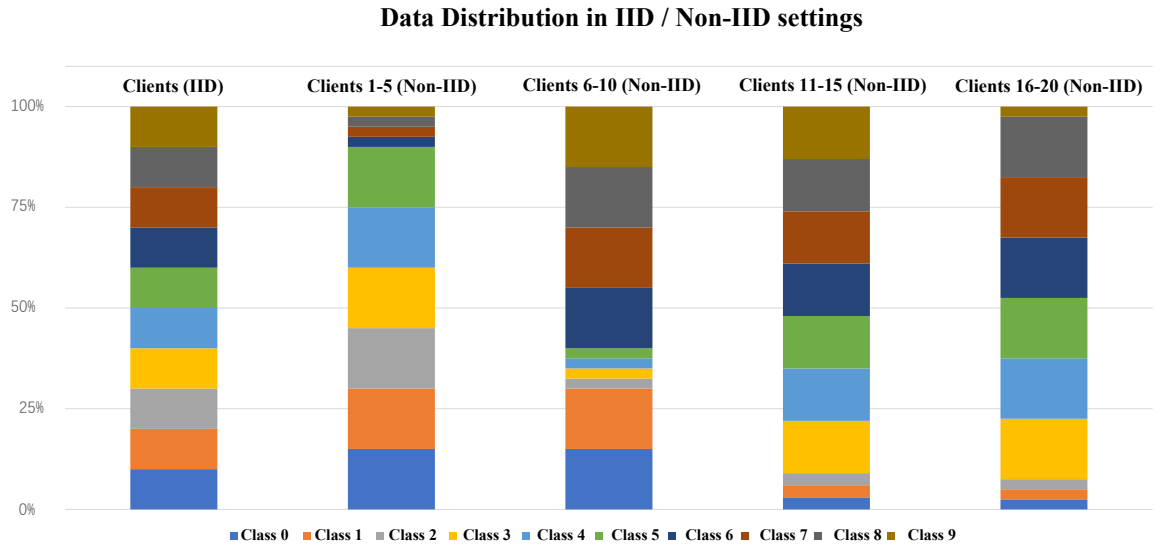

FIGURE 4.1: Illustration of the example of the data distribution (IID setting: left bar, non-IID setting: right four bars).

performance of Vanilla FL [97] in non-adversarial as the best learning performance (upper limit) and demonstrate in the plots of both IID and non-IID settings.

Experimental Results

The results show the existing FL methods (I) can maintain robustness when defense against the Label Flipping Attack in IID scenarios, (II) can not effectively aggregate the information of each client in non-IID scenarios, (III) can not defend against the "Label Flipping Attack" in non-IID scenarios.

(I) The existing FL methods can maintain robustness when defending against the Label Flipping Attack in IID scenarios. First, the existing FL methods achieve a similar global accuracy as Vanilla FL (upper limit). For instance, Vanilla FL achieves 84.5% global accuracy while Krum, Median, Trimmed Mean, and FL Trust achieve 84.43%, 85.22%, 84.47% and 84.25%, respectively. On the other hand, they maintain the accuracy of the attacked class (i.e., class 5). Specifically, class 5 receives 57% testing accuracy under Vanilla FL and receives 56%, 60%, 57%, and 53% in Krum, Median, Trimmed Mean, and FL Trust, respectively. Besides, we note that the information of all classes have effectively been learned by the global

model through the existing FL methods, which receives a similar accuracy compared with the Vanilla FL.

(II) The existing FL methods can not effectively aggregate the information of each client in non-IID scenarios. The result shows that the existing FL methods drop the global accuracy in non-IID scenarios against Label Flipping Attacks". Specifically, Krum, Median, Trimmed Mean, and FL Trust decrease the global accuracy from around 85% to 65%, 66%, 67%, and 73%, respectively. Although some classes have not been attacked, they still decrease testing accuracy (even to 0%), which consequently causes a significant global accuracy decrease. For instance, classes 8 and 9 receive 0% in Krum and Median FL method; they receive 24% and 0% under Trimmed Mean. In contrast, FL Trust maintains a relatively stable accuracy across different classes; only class 8 receives 0% testing accuracy.

(III) The existing FL methods can not defend against the Label Flipping Attack in non-IID scenarios. The existing FL methods failed to maintain the accuracy of the attacked class in non-IID settings. Due to the Label Flipping Attack, class 5 testing accuracy drops to 24% in Trimmed Mean and to 0% in the other three FL methods.

Figure 4.2 and Table 4.1 illustrate the global accuracy and accuracy of different classes of different FL methods against the Label Flipping Attack in IID and non-IID scenarios, respectively. In the next section, we will provide a detailed discussion of the reason pertaining to observed phenomena (II) and (III).

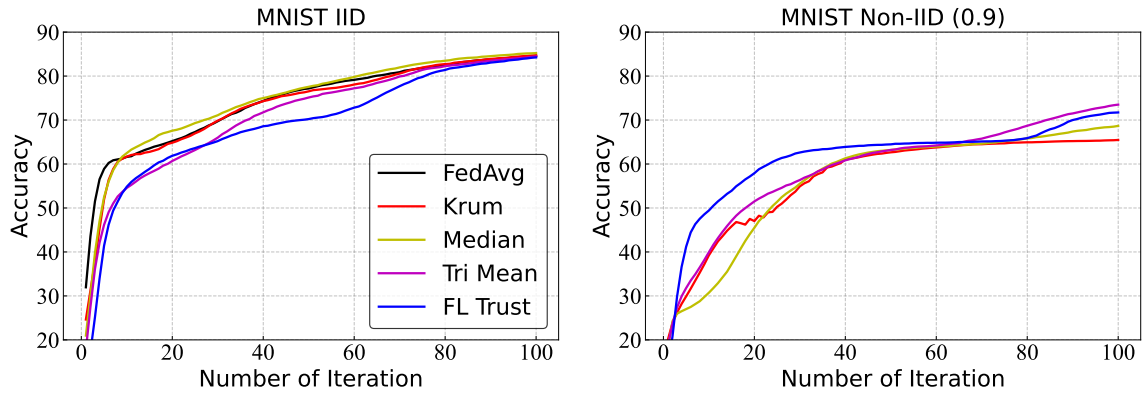

FIGURE 4.2: Illustration of the global accuracy of the existing FL methods against Label Flipping Attack in IID and non-IID scenarios.

TABLE 4.1: Illustration of the classes' accuracy of the existing FL methods against Label Flipping Attack in IID and non-IID scenarios.

Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
FedAvg	96%	95%	89%	88%	85%	60%	92%	87%	79%	82%
$MNIST_{IID}$										
Krum	96%	97%	81%	87%	84%	56%	92%	86%	78%	83%
Median	97%	96%	81%	86%	85%	60%	92%	85%	79%	93%
Tri Mean	96%	96%	81%	87%	77%	57%	92%	86%	81%	83%
FL Trust	95%	96%	75%	88%	81%	53%	91%	86%	81%	93%
$MNIST_{0.9}$										
Krum	95%	97%	83%	92%	94%	0%	92%	81%	0%	0%
Median	92%	96%	80%	92%	94%	2%	95%	87%	0%	0%
Tri Mean	93%	96%	78%	92%	93%	0%	94%	88%	24%	0%
FL Trust	95%	93%	85%	89%	86%	0%	93%	86%	0%	77%

Discussion

In this section, we provide the discussion and answers to two research questions based on the experimental result. (Q1) Why do the existing FL methods fail to aggregate the information of different clients under non-IID scenarios? (Q2) Why do the existing FL methods fail to defend "Label Flipping Attacks?"

A1: First, Krum [9], and Median methods [167] enhance model robustness by selecting the most "honest" gradient (the shortest L2 distance)/coordinate-wise (median value) parameter as the aggregation outcome and updating the global model. As the client owns class-balanced training data in the IID setting proposed, one client gradient can evenly carry all class information which enables the global model to increase accuracy and collect information on each class. However, in non-IID settings, especially when the class imbalance exists, the client gradient may be biased and majorly carry information of the biased class. Directly using one client's gradient as the aggregation outcome in such cases may lead the global model to achieve good learning performance on the biased classes but put the minor classes (e.g., Class 8, 9 in the example) at disadvantages. Second, the Trimmed Mean [167] enhances the model

robustness by trimming the extremum of each parameter. Unlike model poisoning attacks that directly scale each parameter, the Label Flipping Attack generates the crafted gradient through poisoning training data. As the attack strategy introduces a smaller crafting magnitude (compared with model poisoning attack), trimming extremum may not effectively remove adversarial parameters but remove the parameter from benign clients instead, which leads the global model to lose information (i.e., Class 8 and 9) in aggregation and achieve a decreased accuracy. Furthermore, the local model performance-based robust aggregation methods, including FL Trust, are relatively fixed when generating the aggregation and assigning weights. For instance, the cosine similarity between the client's and severe gradient is highly related to the data distribution and quality of the client. The client owning IID-like and high-quality training data can continuously receive a high cosine similarity and heavyweight in FL Trust [16] aggregation unless it changes the data distribution pattern. In contrast, other clients will hardly participate in the aggregation (or receive a small weight), putting the global model in the difficulty that collecting information from these clients.

A2: To successfully perform the Label Flipping Attack and effectively reduce the accuracy of the targeted class, the malicious clients (and the corresponding gradients) should keep participating in the learning task; once absent, the reduced accuracy will quickly return to the baseline [136]. In other words, in the proposed example, the attackers are excluded from the aggregation in the IID setting but can keep participating in the learning task of the non-IID setting under the existing FL methods. We consider this because even the benign gradients are relatively heterogeneous in the non-IID setting, which drives difficulty in identifying the crafted gradient. Specifically, the existing FL methods regard the outliers (gradients or parameters) as adversarial and enhance model robustness by excluding these potential malicious gradients or parameters from the aggregation. In IID scenarios, clients' gradients tend to be similar; the scaled gradient can be easily identified and excluded by the existing FL methods. In contrast, even benign clients' gradients could be very different in the non-IID setting. As the gradient generated by the Label Flipping Attack carries the right information of most (uncrafted) classes, it can mingle in the benign gradients, which brings difficulty for the existing FL methods.

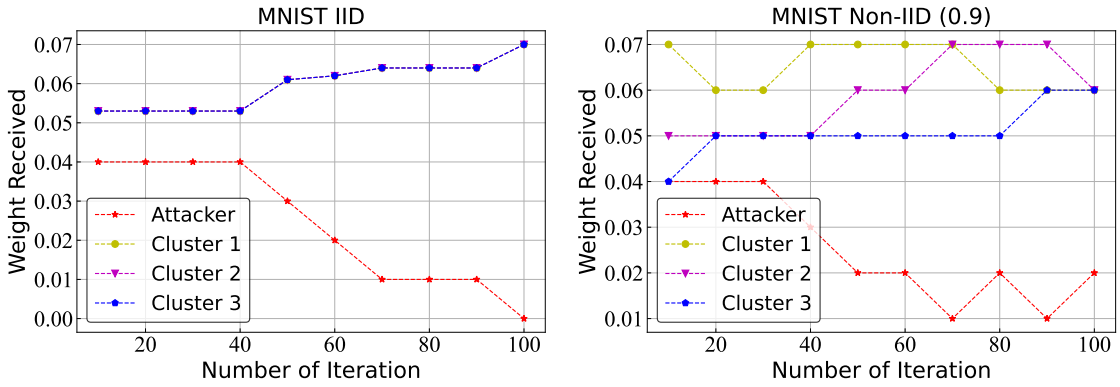

FIGURE 4.3: Illustration of the weight assigned by each client under the FL Trust method against "Label Flipping Attack" in IID and non-IID scenarios.

We provide an example for monitoring the weight of each client cluster (refer to the data distribution mentioned in the previous Experimental Setup) per 10 iterations in aggregation and note that the crafted gradients can keep participating in the aggregation even within the existing FL Trust. Figure 4.3 illustrates the weight assigned to each client cluster in the FL Trust. As there are four different data distributions (four client clusters) in the non-IID setting, we monitor the weight assigned by each client with different data distributions in each 10 learning rounds. The results show that the attacker (client cluster 1) receives a decreasing aggregation weight within the whole learning task in the IID setting and is excluded from aggregation since iteration 90 (i.e., receives weight 0%). In contrast, attackers keep participating in the learning task although witnessing a decreasing weight; at the 100th iteration, each attacker still receives 2% weight, which makes up 10% in total. Besides, we note that although client clusters 2, 3, and 4 are all benign, they receive unequal weight in the aggregation, which explains why Class $1 \sim 10$ can have different learning performances.

4.1.4 Honest Score Client Selection based Federated Learning

4.1.4.1 Overview

To enhance the robustness of the FL system, we propose the Honest Score Client Selection scheme (short as HSCS) and corresponding FL framework (short as HSCSFL). Instead of eliminating the impact of the adversary during the aggregation, HSCSFL considers only

selecting top $p\%$ contribution score clients to participate in the learning task and excludes the adversary before the aggregation; here, $p\%$ is a preset parameter base on the knowledge. In the next section, we will discuss why the gradient generated by the Label Flipping Attack achieves a lower contribution score compared with the benign gradients and how the contribution score client selection scheme works. The objectives of HSCSFL are shown as the following:

(I) HSCSFL should effectively aggregate the information of different classes in non-IID scenarios; in other words, classes should achieve a similar learning performance.
(II) HSCSFL should be robust when defending against Label Flipping Attack; the attacked class should keep the accuracy similar to the non-adversary level.

4.1.4.2 Assumption

In HSCSFL, we consider the server to be honest and act as a defender. The server should be able to collect a clean evaluation dataset that evenly includes each class. We keep the assumption set in Krum [9] that the server knows the number (i.e., $p\%$) of malicious clients. We consider both IID and non-IID scenarios and agree that the marginal distribution deference is the main non-IID resource. We follow a general setting in which all clients (i.e., benign and adversary clients) keep relatively similar diversity and volume training data.

4.1.4.3 Honest Score Client Selection Scheme

Honest Score Client Selection scheme framework

We consider a dynamic client selection scheme to select benign participants based on the current global model and exclude potential malicious clients before aggregation. As the key for attackers reducing target class accuracy through Label Flipping Attack is participating in the aggregation of each iteration [136], the proposed scheme should notice immediately once the model has been attacked and exclude the potential adversary in the next few learning rounds. We consider the server to play the defender's role and can collect a small, clean evaluation dataset to monitor the global model. As the "Label Flipping Attack" only decreases the accuracy of the attacked class, evaluating the overall performance of client gradients is not enough and can not identify the crafted gradients effectively. Thus, our contribution score

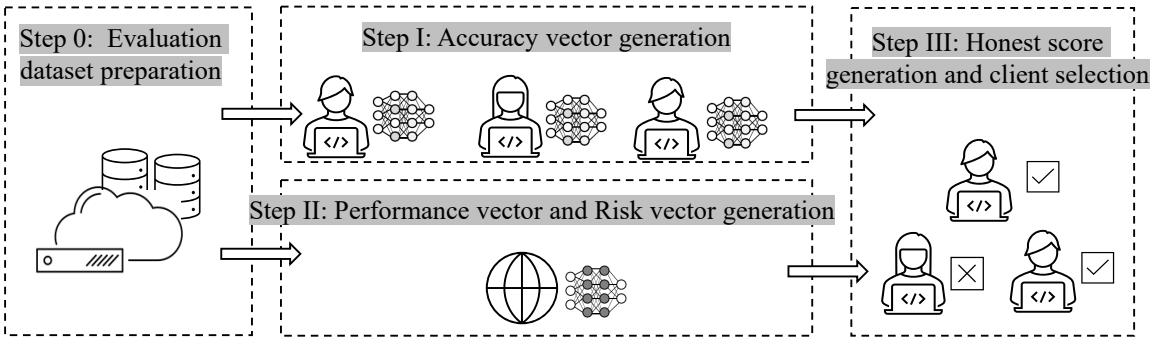

FIGURE 4.4: Illustration of the framework of the Honest Score Client Selection (HSCS) scheme.

client selection evaluates each class separately and generates the corresponding accuracy vector. We further monitor the accuracy of each class of the global model through the evaluation dataset and generate the potential risk victor; the class that achieves a low accuracy will be assigned a high-risk value. The server finally times clients' accuracy vector and the global model's risk vector to generate the honest score for all clients; only top $p\%$ clients can participate in the following aggregation. By leveraging the risk vector, the server can effectively notice if a class has been attacked and flag the class with a high risk value. Adversaries that attend to attack that class will receive a lower honest score than other benign clients in the next learning round, which can not further participate in the aggregation.

Specifically, honest score client selection includes the following four steps. Figure 4.4 illustrates the framework of the Honest Score Client Selection scheme.

- Step 0: Evaluation dataset preparation

  To identify the malicious gradients, HSCS requires the server to collect a small and clean evaluation dataset before the learning task starts. This evaluation dataset should evenly include each class data. In this study, we set the evaluation data size as 5% of the overall training data volume. Following the research [16, 110], we consider the evaluation dataset could be collected manually.

- Step I: Accuracy vector generation

  Once a preset number of clients' gradients have been received, the server evaluates the local model updates through the evaluation dataset. Different from FL

Trust and Sageflow evaluate the local model updates from the overall perspective, HSCS evaluates the clients' gradients from each class. At the end of Step I, all local updates should be evaluated, and each of them has an "accuracy vector" (represented as $AccV$) that records its performance (i.e., accuracy) on each class.

- Step II: Performance vector and Risk vector generation

  To identify the class that could potentially be attacked and avoid being further affected, we monitor the accuracy of each class of the global model and flag the lower accuracy class with a higher risk value. Specifically, the server first evaluates the global model at this round and generates the "global model performance vector" (represented as $PerV$)which includes each class's accuracy. Then, the server generates the "Risk vector" (represented as $RisV$) by the following equation:

$$RisV = 1 - PerV \tag{4.5}$$

  At the end of Step II, the class with high accuracy of the global model is assigned with a low risk value and vice versa; all risk values are recorded in the $RisV$.

- Step III: Honest score generation and client selection

  Finally, the server generates the honest score (represented as HS) for each client based on the $AccV$ and $RisV$ by equation 4.6, where i indicates the ith client, C denotes the number of class. $AccV$ and $RisV$ include c elements respectively.

$$HS_i = \sum_{j=1}^{C} AccV_i * RisV \tag{4.6}$$

  The server subsequently selects the top p % HS clients to participate in the following aggregation. The $p\%$ is a preset parameter designed based on the knowledge of the amount of the adversary.

Toy example

In this section, we provide a toy example to further demonstrate and clarify how the HSCS scheme works. We consider a simple 3 classes ($c = 3$) FL scenario with a server and 4 clients ($n = 4$). Suppose an evaluation dataset that evenly includes 3 classes has already been prepared. At the tth learning round, the server first evaluates 4 clients' gradients through

the evaluation dataset and generates the corresponding $AccV$. Suppose the clients' $AccV$ is generated as the following:

$$AccV_1 = [0.71, 0.82, 0.65],\ ...,\ AccV_4 = [0.41, 0.80, 0.97] \tag{4.7}$$

Then, the server evaluates the current global model by the evaluation dataset to generate the $PerV$ and corresponding $RisV$. Suppose the $PerV$ and $RisV$ are generated as the following:

$$PerV = [0.24, 0.55, 0.57],\ RisV = 1 - Per = [0.76, 0.45, 0.43] \tag{4.8}$$

The server generates the HS for each client base on their $AccV$ and the $RisV$. Here, we demonstrate the process for generating HS_1:

$$HS_1 = \sum_{j=1}^{3} AccV_1 * RisV = 0.71 * 0.76 + 0.82 * 0.45 + 0.65 * 0.43 = 1.19 \tag{4.9}$$

Finally, top $p\%$ HS clients will be selected and participate in the following aggregation. Suppose $p = 50\%$, Hs_1, and HS_2 are the top two scores; clients 1 and 2 will be consequently selected in aggregation.

This toy example also shows the importance and benefits of evaluating the performance of the client's gradient in each class. Specifically, class 1 achieves a significantly lower accuracy on the current global model; compared with other classes, class 1 is highly likely to be attacked. When evaluating clients from the overall perspective (like the existing FL works), client 1 and client 4 are difficult to compare as they achieve a similar performance (mean accuracy as 72%). However, as class 1 has a high risk of being attacked, Client 4 is more suspicious in the current iteration, which should be avoided to be selected. Thus, the HSCS scheme assigns client 4 a lower HS (HS_4 = 1.0887) than client 1 (HS_1 = 1.19), which can effectively exclude client 4 from the subsequent aggregation.

4.1.4.4 Honest Score Client Selection Scheme based Federated Learning

The Honest Score Client Selection scheme based on Federated Learning (short as HSCSFL) follows the standard (Vanilla) FL [97] and introduces the HSCS before the aggregation stage to mitigate "Label Flipping Attack." Specifically, HSCSFL includes the following five stages (i.e., Global Model Broadcast, Local Training, Gradients Upload, Client Selection, Aggregation), Figure 4.5 illustrates the HSCSFL framework.

- **Stage I: Global Model Broadcast**

  At each learning iteration, the server first broadcasts the current global model to all participants and potential clients. In the first communication round of the learning task, the server clarifies the learning objective and rewards to attract the participants.
- **Stage II: Local Training**

  After receiving the current global model from the server, each client locally trains the model through the privacy data.
- **Stage III: Gradients Upload**

  Once the preset model accuracy or training round reached, all clients send the local model updates back to the server.
- **Stage IV: Client Selection**

  Based on the clients' gradient received, the server performs HSCS scheme to select $p\%$ client to participate in subsequent aggregation. Specifically, the server first generates the Accuracy Vector for all clients and the Risk Vector of the current global model. Then, the server calculates and ranks the Honest Score for all participants to select aggregation candidates.
- **Stage V: Aggregation**

  Finally, the server aggregates the $p\%$ gradients by averaging. The aggregation outcome is subsequently used to updates the global model to finish this learning iteration. Formula 4.10 illustrates the aggregation rule of HSCSFL.

$$M^{t+1} = M^t + \eta \cdot \frac{1}{p} \sum_{i=1}^{p} g_i^t,\ p = p\% \cdot n \tag{4.10}$$

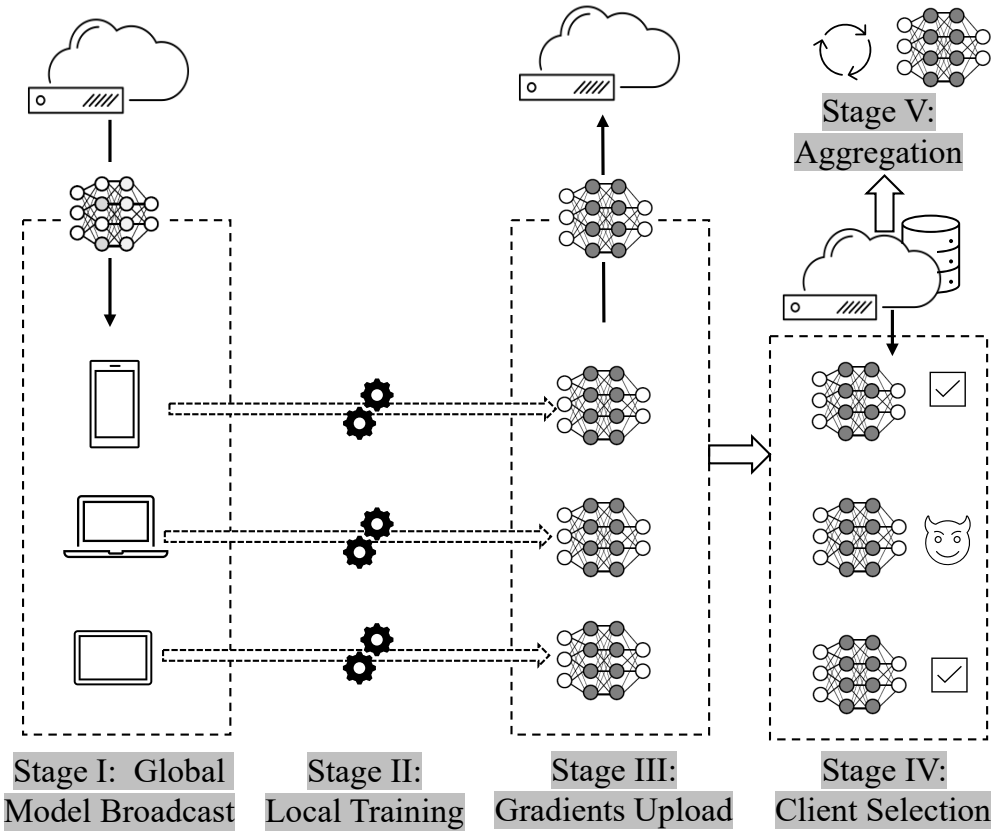

FIGURE 4.5: Illustration of the HSCSFL framework.

4.1.5 Evaluation

In this section, we comprehensively evaluate the robustness of HSCSFL when defending against Label Flipping Attack. We consider different non-IID degrees and attack strategies in the evaluation.

4.1.5.1 Dataset

Expect the MNIST dataset introduced in Section 4.1.3, we further introduce the Fashion MNIST dataset [152] for evaluation. Fashion MNIST (short as FMNIST) consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 gray-scale image associated with a label from 10 classes of fashion items. To simulate different non-IID scenarios in the real world, we set the non-IID degree as 0.8 and 0.9 to generate $MNIST_{0.8}$,

$MNIST_{0.9}$, and $FMNIST_{0.8}$, $FMNIST_{0.9}$. We follow the data distribution introduced in Section 4.1.3 to distribute data to 20 clients. We consider the server collects/owns a clean evaluation dataset to support performing the HSCS scheme.

4.1.5.2 Benchmark

In our experiments, we utilize the representative byzantine-resilient FL methods introduced in Section 4.1.3 as benchmarks. Additionally, we include the performance results of FedAvg under the IID setting. We note that the testing performance of FedAvg (Vanilla FL) with IID serves as an indicator of the potential upper bound for class or global accuracy but cannot be directly compared with other benchmarks due to differences in training data distributions. The inclusion of FedAvg with the IID setting is aimed at demonstrating how our HSCSFL approach can achieve the upper bound of learning performance in various scenarios.

4.1.5.3 Threat Model

We consider 25% clients have been controlled by the adversary; the attackers can generate the crafted gradient through the Label Flipping Attack based on the data owned. We consider the adversary can select different attacking strategies. Table 4.2 illustrates the different attack strategies of the Label Flipping Attack simulated in the experiments. Specifically, we simulate the adversary attacks on the most vulnerable class (class 5 in MNIST), normal class (class 5 in FMNIST), and multiple classes (class 0 and 1 in FMNIST).

We also consider an adaptive adversary environment, which includes 15% and 35% overall attackers. We leave the adaptive experiments and the corresponding discussion in subsection 4.1.5.6.

TABLE 4.2: Illustration of the different attack strategies of "Label Flipping Attack."

Strategy	Dataset	Adversary Amount	Original Class	Target Class
1	MNIST	25%	5	8
2	Fashion MNIST	25%	5	8
3	Fashion MNIST	25%	0, 1	8

TABLE 4.3: Illustration of the global accuracy under different attack strategies of "Label Flipping Attack."

Dataset	Krum	Median	Trimmed Mean	FL Trust	HSCSFL
Atatck Strategy 1					
$MNIST_{0.8}$	65.49%	68.66%	73.50%	73.7.%	87.16%
$MNIST_{0.9}$	65.04%	65.71%	67.22%	72.45%	88.25%
Attack Strategy 2					
$FashionMNIST_{0.8}$	67.59%	62.13%	70.51%	69.11%	76.73%
$FashionMNIST_{0.9}$	65.07%	56.40%	68.55%	60.21%	75.64%
Attack Strategy 3					
$FashionMNIST_{0.8}$	73.43%	65.39%	72.54%	70.01%	76.74%
$FashionMNIST_{0.9}$	72.67%	62.59%	66.61%	59.31%	77.20%

4.1.5.4 Experimental Results

The experimental results show our HSCSFL effectively enhances the model robustness and maintains global accuracy when defending the Label Flipping Attacks with different attack strategies. Recall the objectives set in Section 4.1.4 are (I) effectively aggregate the information of different classes, and (II) effectively defend against Label Flipping Attacks in non-IID settings; our HSCSFL achieves both two objectives. Table 4.3, Figure 4.6, and 4.7 compare the global accuracy of the different FL methods against Label Flipping Attacks in the FMINST and MNIST datasets with different non-IID scenarios.

First, HSCSFL effectively aggregates the information of different classes in different non-IID settings and maintains global accuracy against the Label Flipping Attack. Vanilla FL [97] achieves 84.5% and 77.64% in MNIST and FMNIST, while HSCSFL witnessed a similar testing accuracy, which achieves 87.16%, 88.25% in MNIST under attack strategy 1, 76.73%, 75.64% in FMNIST under attack strategy 2. When two FMNIST labels are attacked, HSCSFL still maintains a global accuracy of around 76.50%. In contrast, the existing FL methods show a significantly degraded performance. For instance, Krum [9] and Median [167] decrease the global accuracy to around 65% in two datasets when the non-IID degree is 0.8 and maintain

the global accuracy when increasing the non-IID degree as 0.9. On the other hand, Trimmed Mean [167] and FL Trust [16] achieve a higher accuracy (70%+) under two datasets when the non-IID degree is 0.8 but witness a decreasing model performance when increasing the non-IID degree. The global accuracy comparison shows our HSCSFL can effectively aggregate the client's model updates and learn information from different classes in various non-IID scenarios; we discuss the detailed accuracy of each class in the next paragraph.

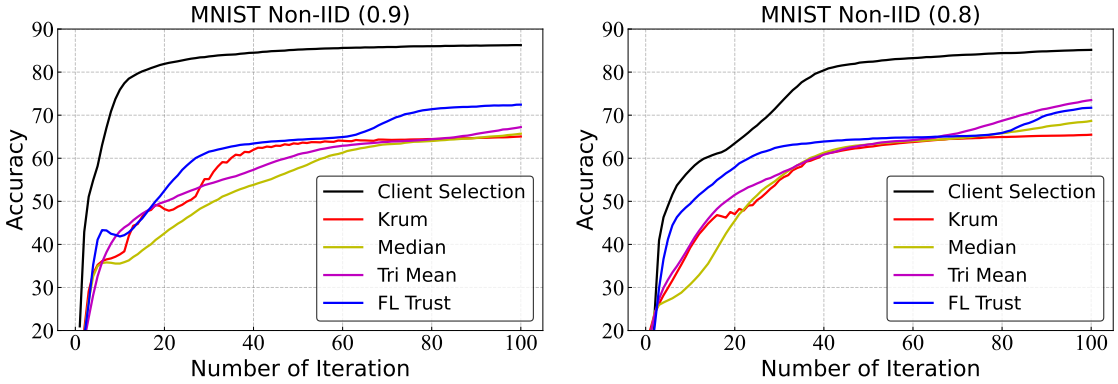

FIGURE 4.6: Illustration of the global accuracy of the different FL methods against "Label Flipping Attack" (attack strategy 1) in MNIST with non-IID scenarios.

TABLE 4.4: Illustration of the classes' accuracy of the existing FL methods against "Label Flipping Attack" in IID and non-IID scenarios.

$MNIST_{0.9}$										
Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
FedAvg	96%	95%	89%	88%	85%	57%	92%	87%	79%	82%
HSCSFL	96%	83%	88%	89%	88%	66%	93%	90%	87%	88%

Second, HSCSFL effectively defends against Label Flipping Attacks and maintains the attacked class accuracy in non-IID settings. Specifically, HSCSFL achieves 66%, 65%, 77%, and 78% accuracy on the attacked class in $MNIST_{0.9}$ and $MNIST_{0.8}$ under attack strategy 1, $FMNIST_{0.9}$ and $FMNIST_{0.8}$ under strategy 2 respectively; which is similar to, even higher to the upper bound carried by the FedAvg. On the contrary, almost all existing FL methods fail to maintain the model robustness and receive significant accuracy reduction on the attacked class. Krum shows the worst performance when defending against Label

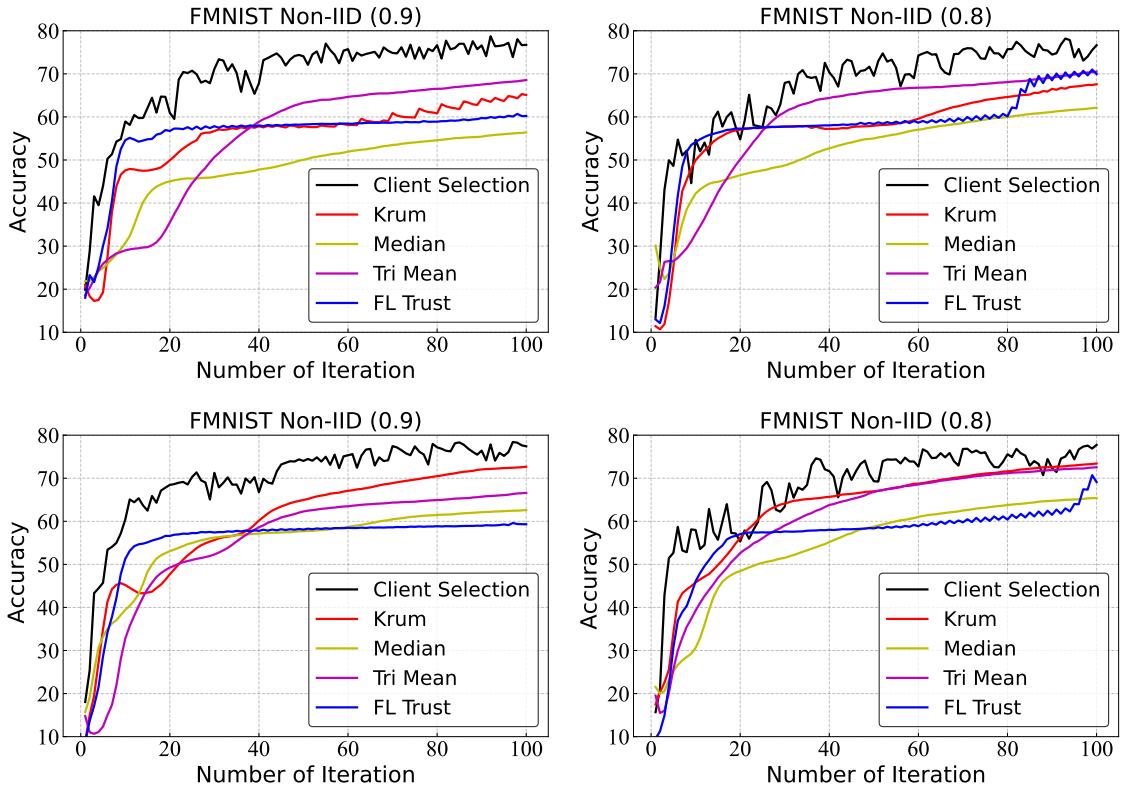

FIGURE 4.7: Illustration of the global accuracy of the different FL methods against "Label Flipping Attack" with attack strategy 2 (up two), with strategy 3 (down two) in Fashion MNIST with non-IID scenarios.

Flipping Attacks and receives 0% class accuracy on almost all datasets. Under attack strategy 3, HSCSFL also achieves a good defending performance, which witnesses around 80% and 92% on two attacked classes.

Besides, the unattacked classes have generally not been affected under HSCSFL. Unlike the existing FL methods tend to select one most honest client's gradient/parameter as the aggregation outcome, HSCSFL averages each client's gradient after performing the HSCS scheme, which enables their carried information can be equally learned by the global model. For instance, although class 8 has not been attacked in $FMNIST_{0.9}$, Krum [9], Median [167], Trimmed Mean [167], and FL Trust[16] reduce the class accuracy to 55%, 33%, 84%and 0%, respectively; while HSCSFL achieves 90% accuracy. Table 4.4, 4.5, and 4.6 illustrate the detailed class accuracy of different FL methods under MNIST and FMNIST. For the sake of

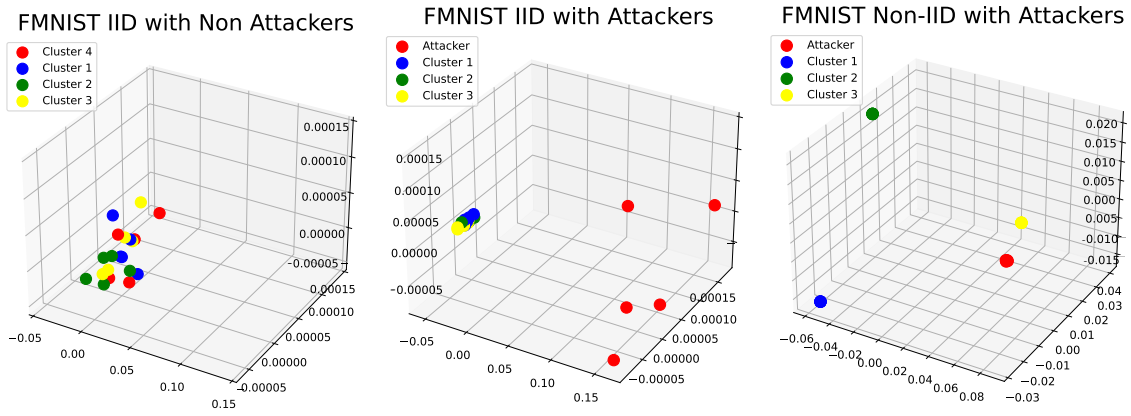

FIGURE 4.8: Illustration of the clustering outcome with Fashion MNIST in non-adversarial under IID setting, adversarial under IID setting, and adversarial under Non-IID setting.

brevity, Table 4.4 only includes the accuracy information of HSCSFL; the full experimental information can be found in Section 4.1.3.

TABLE 4.5: Illustration of the classes' accuracy of different FL methods against Label Flipping Attacks in MNIST with non-IID setting.

Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
					$MNIST_{0.8}$					
FedAvg	96%	95%	89%	88%	85%	57%	92%	87%	79%	82%
Krum	96%	97%	84%	94%	95%	0%	92%	83%	0%	0.1%
Median	94%	95%	81%	92%	94%	38%	95%	87%	0%	0%
Tri Mean	93%	95%	81%	91%	94%	34%	94%	88%	54%	0%
FL Trust	95%	95%	84%	90%	89%	0%	94%	88%	0%	65%
HSCSFL	97%	97%	85%	88%	92%	65%	93%	89%	79%	80%

4.1.5.5 Effectiveness of clustering method

We note that the existing work [80] shows that clustering method could be introduced at the server to distinguish malicious gradients (under Label Flipping Attack) from benign clients in IID scenarios. However, whether the clustering method also shows effectiveness in Non-IID settings remains unclear. In this section, we investigate the effectiveness of the clustering method in IID with non-adversarial, IID with Label Flipping Attackers, and Non-IID with

TABLE 4.6: Illustration of the classes' accuracy of different FL methods against Label Flipping Attacks in FMNIST non-IID settings.

Class	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
FadAvg	77%	92%	64%	84%	76%	66%	40%	87%	92%	94%
$FMNIST_{0.9}$										
Krum	76%	89%	50%	87%	84%	0.4%	24%	91%	55%	92%
Median	70%	90%	42%	85%	77%	23%	48%	92%	33%	0%
Tri Mean	72%	89%	42%	86%	76%	10%	42%	93%	84%	86%
FL Trust	83%	91%	68%	85%	78%	11%	0.7%	90%	0%	93%
HSCSFL	78%	92%	41%	80%	89%	77%	48%	94%	90%	85%
HSCSFL(S3)	83%	91%	69%	81%	86%	77%	25%	94%	91%	85%
$FMNIST_{0.8}$										
Krum	69%	90%	41%	86%	78%	0%	48%	91%	80%	92%
Median	73%	91%	47%	85%	80%	34%	43%	95%	71%	0%
Tri Mean	74%	90%	48%	85%	79%	20%	35%	92%	88%	94%
FL Trust	81%	90%	64%	85%	81%	23%	6%	92%	83%	92%
HSCSFL	85%	92%	94%	87%	66%	78%	23%	92%	93%	87%
HSCSFL(S3)	86%	92%	53%	80%	93%	78%	17%	93%	93%	87%

Label Flipping Attackers. We follow the scenario set before, use the $FMNIST_{0.9}$ as the a training data, attacking strategy 3 for performing attack and PCA (Principal Component Analysis) [80] as the clustering method. To provide more information, we set the principal Component as 3.

Figure 4.8 illustrates the clustering results for three different scenarios. In an IID setting with a non-adversarial scenario, participants (or gradients) are close due to the similar information carried by the training data. However, when Label Flipping Attacks are introduced in this IID setting, benign clients tend to maintain their close pattern. In contrast, the gradients of malicious clients deviate significantly, creating a noticeable distance from the benign gradients. This makes it feasible to effectively distinguish between them [80]. However, due to the distinct information they carry, even benign clients exhibit varied behaviors and maintain

significant distances from one another. This variation makes it challenging to distinguish malicious clients without additional information.

4.1.5.6 Discussion

In this subsection, we provide the discussion around HS difference between benign and malicious clients, and the performance of different FL methods (including the existing FL methods and our proposed HSCSFL) under various adversarial environments.

A) HS Between Benign and Malicious Clients

HSCSFL maintains the model robustness through the HSCS scheme, which assigns the suspicious clients lower scores and excludes them from the aggregation. Thus, ensuring the adversary achieves a lower honest score than the benign clients is the key to effectively performing HSCSFL. As the adversary generates the crafted gradient based on the same data distribution and volume (as shown in Section 4.1.3) in this paper, they generally receive the same honest score (HS) in each iteration. We monitor the honest score difference (HS_d) between benign clients and adversaries; specifically, the positive HS_d indicates the attackers achieve lower HS than the benign client and vice versa.

Figure 4.9 illustrates the HS_d between benign clients and malicious clients in FMNIST with different non-IID degrees. The results show that benign clients can generally keep a higher HS than the adversary during the learning process; in other words, the adversary can hardly participate in the aggregation and perform the attack. For instance, the adversary only participates in the aggregation two times under FMNIST when the non-IID degree is 0.8 and attack strategy 2 is performed. Recall the necessary condition for successfully performing a Label Flipping Attack includes the adversary should keep participating in the aggregation [136], our HSCSFL can effectively defend against Label Flipping Attack and achieve a good learning performance. We consider this HS gap to be "unavoidable" for performing the Label Flipping Attack. From the HSCS scheme perspective, the clients' gradients' performance loss in each class will be counted and collected as the $AccV$. As Our HSCSFL weights the class's accuracy $AccV$ dynamically based on the global model, the accuracy gap on the potential attacked class is scaled up, which results in the adversary achieving a lower HS compared

with the benign clients. Next, we provide a formal discussion of the organization of the "HS gap" and the effectiveness of the HSCS scheme.

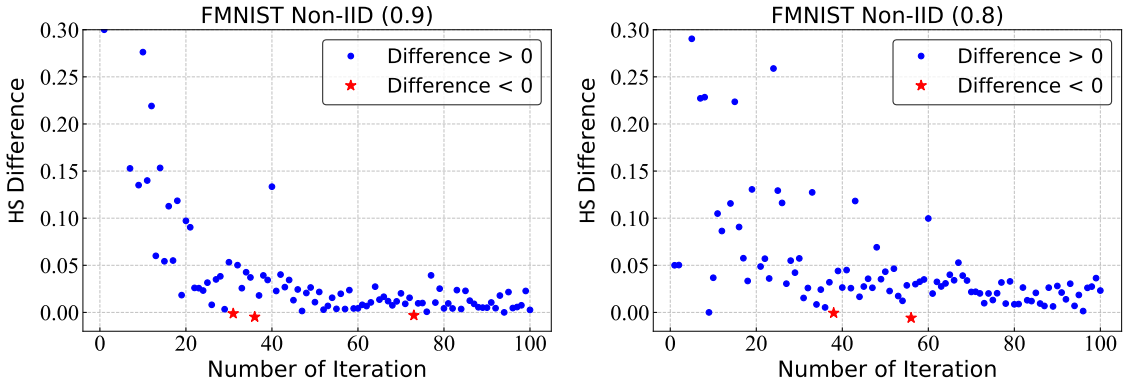

FIGURE 4.9: Illustration of the HS difference between adversary and benign clients in FMNIST (attack strategy 2) with different attack strategies. The blue point indicates the benign clients achieve a higher HS and the difference is positive; the red star indicates the benign clients achieve a smaller HS and the difference is negative.

Suppose the current learning task includes C classes in total. The Label Flipping Attackers select the class m data to poison and generate the corresponding malicious gradients g^*, while the normal clients generate the benign gradients g. Suppose the adversary effectively attacks the global model and reduces the accuracy of class m in iteration t; we have the following:

$$PerV^t(m) < PerV^t(c), c \in C, c \neq m \tag{4.11}$$

$$RisV = 1 - PerV, RisV^t(m) > RisV^t(c), c \in C, c \neq m \tag{4.12}$$

To select the most honest clients to participate in $t+1$ round aggregation, the server calculates the HS for each client. The HS includes the HS for benign classes (c) and attacked class (m) which could be further divided by the following:

$$HS = HS(m) + HS(c), c \in C, c \neq m \tag{4.13}$$

Based on the equation 4.6 and 4.13, the HS for the adversary and benign clients are shown as equation 4.14 and 4.15, respectively.

$$HS_{g^*}^{t+1} = AccV_{g^*}^{t+1}(m) \cdot RisV^t(m) + AccV_{g^*}^{t+1}(c) \cdot RisV^t(c) \tag{4.14}$$

$$HS_{g}^{t+1} = AccV_{g}^{t+1}(m) \cdot RisV^t(m) + AccV_{g}^{t+1}(c) \cdot RisV^t(c) \tag{4.15}$$

Recall we consider the adversary and benign clients holding similar capability and quality training data (see Section 4.1.3), we have the following:

$$AccV_{g^*}^{t+1}(c) \cdot RisV^t(c) \simeq AccV_{g}^{t+1}(c) \cdot RisV^t(c) \tag{4.16}$$

$$Hs_{g}^{t+1} - HS_{g^*}^{t+1} \simeq RisV^t(m) \cdot (AccV_{g}^{t+1}(m) - AccV_{g^*}^{t+1}(m)) \tag{4.17}$$

Following formula 4.17, we can find the HS gap between the benign clients and the adversary is the accuracy difference of their model performed on the attacked class (m) and then large-scaled by $RisV(m)$. The "large" we mentioned here is compared with the $RisV$ of other unattacked classes ($RisV(c)$), which is derived by 4.12. As long as the adversary performs Label Flipping Attacks, its model unavoidably achieves lower accuracy on the class m and is further large-scaled, resulting in a lower HS (we called HS gap) compared to benign clients, shown in Figure 4.9.

B) Performance of FL Methods in Various Adversarial Environments

We consider the adversary could occupy 15%, 25%, and 35% overall clients and further compare the existing FL methods and our proposed HSCSFL on FMNIST with a non-IID degree of 0.8 when attack strategy 2 is performed. Table 4.7 illustrates the testing accuracy of the attacked class under different FL methods and adversary occupations.

The experimental results show that the existing FL methods drop the learning performance on the attacked class when increasing the occupation of Label Flipping Attack adversaries, while HSCSFL can effectively defend against the attack. Specifically, Krum and Trimmed Mean achieve around 0% when the amount of adversary reaches 25%. FL Trust shows a better

defense performance when including more attackers while receiving around 36% class testing accuracy under 15% adversary and 24% under 25% and 35% adversary.

TABLE 4.7: Illustration of the attacked class's accuracy under different FL methods against Label Flipping Attacks in FMNIST with non-IID degree as 0.8 when 15%, 25%, and 35% clients are adversarial.

Adversary Percentage	**Krum**	**Median**	**Trimmed Mean**	**FL Trust**	**HSCSFL**
15%Adversary	53.7%	53.9%	28.9%	36.6%	71.1%
25%Adversary	0%	34.5%	0.8%	23.4%	78.0%
35%Adversary	0.8%	0%	4.1%	23.9%	74.6%

C) Evaluation Dataset

In HSCSFL we consider the evaluation dataset has been collected manually by the server (service provider) to guarantee its clean and unbiased. Here, we provide the discussion around the impact once the evaluation dataset has been polluted by an unknown attack or invasion.

We begin by formally expressing the Honest Score for both benign and malicious clients, where the evaluation dataset is biased or compromised. Following the formulas 4.14 and 4.15, we reorganize the Honest Score as unaffected and affected to reflect the evaluation dataset of the corresponding classes are biased or polluted and clean, respectively.

$$HS = HS_{unaff} + HS_{aff} \tag{4.18}$$

Equation 4.18 could be extended as 4.19 and 4.20 for benign and malicious clients, respectively. We use $c \in C - C^*$ to denote the class that are clean in the evaluation dataset, and use $c^* \in C^*$ to denote the polluted or biased class.

$$HS_g^{t+1} = \sum_{c=1}^{C-C^*} HS(c)_g^{t+1} + \sum_{c^*=1}^{C^*} HS(c^*)_g^{t+1}, c^* \neq c \tag{4.19}$$

$$HS_{g^*}^{t+1} = \sum_{c=1}^{C-C^*} HS(c)_{g^*}^{t+1} + \sum_{c^*=1}^{C^*} HS(c^*)_{g^*}^{t+1}, c^* \neq c \tag{4.20}$$

From equations 4.19 and 4.20 we can find that whether the attacked class c belongs to clean classes ($C - C^*$) or biased/polluted classes (C^*) of the evaluation dataset will drive different outcomes, we further discusses these two situation separately.

Situation I: When attacked class $m \in C - C^*$, we have the following:

$$\sum_{c=1}^{C-C^*} HS(c)_g^{t+1} > \sum_{c=1}^{C-C^*} HS(c)_{g^*}^{t+1} \tag{4.21}$$

On the other hand, both benign and malicious clients receive the similar magnitude carried by the bias of class c^*. Thus, we have the following:

$$\sum_{c^*=1}^{C^*} HS(c^*)_g^{t+1} \simeq \sum_{c^*=1}^{C^*} HS(c^*)_{g^*}^{t+1} \tag{4.22}$$

Through combining 4.21 and 4.22, we have the following:

$$\begin{gathered} \sum_{c=1}^{C-C^*} HS(c)_g^{t+1} + \sum_{c^*=1}^{C^*} HS(c^*)_g^{t+1} > \sum_{c=1}^{C-C^*} HS(c)_{g^*}^{t+1} + \sum_{c^*=1}^{C^*} HS(c^*)_{g^*}^{t+1} \\ HS_g^{t+1} > HS_{g^*}^{t+1} \end{gathered} \tag{4.23}$$

Consequently, even when the evaluation dataset is biased or polluted, our HSCSFL algorithm maintains its robustness and effectively filters out the malicious clients, provided that the attacked classes fall within the clean/unbiased classes in the evaluation dataset.

Situation II: When attacked class $m \in C^*$, the inequality 4.21 will represented as the following:

$$\sum_{c=1}^{C-C^*} HS(c)_g^{t+1} \simeq \sum_{c=1}^{C-C^*} HS(c)_{g^*}^{t+1} \tag{4.24}$$

However, since the evaluation data for the attacked class m has been polluted or biased, false negatives and false positives will adversely affect the corresponding score. As a result, the relationship between $\sum_{c^*=1}^{C^*} HS(c^*)_g^{t+1}$ and $\sum_{c^*=1}^{C^*} HS(c^*)_{g^*}^{t+1}$ becomes uncertain, potentially leading to situations where malicious clients achieve a higher Honest Score (HS).

In summary, while HSCSFL's robustness may only be compromised in scenarios where the evaluation data for the attacked class m is polluted or biased, we consider that such intrusive tampering with the evaluation is impractical in real-world scenarios. This is due to

the evaluation dataset being meticulously prepared and securely maintained by the server or service provider, which is honest and plays the defender role.

4.1.6 Conclusion

In this chapter, we evaluate the robustness of the existing FL methods when defending against Label Flipping Attacks in both IID and non-IID scenarios. We find that although these FL methods maintain robustness in IID settings, they show a degrading performance and can not effectively aggregate different class information in non-IID scenarios. To defend against the Label Flipping Attacks and achieve byzantine robustness, we propose the HSCS scheme, which enables the server to select the most honest clients and excludes the suspicious from the aggregation. We further provide the corresponding HSCSFL, and our evaluation shows that the HSCSFL can effectively select benign clients, aggregate the gradients' information from benign clients, and defend against the "Label Flipping Attack" in different non-IID scenarios.

4.2 Preventing Class Imbalance Attack

4.2.1 Introduction

Federated Learning (FL) [97] has emerged as a groundbreaking approach that allows multiple clients to collaboratively train a machine learning model without compromising the privacy of their raw data. Today, FL has demonstrated successful deployment across various domains, including next-world prediction on mobile phones [59], vehicle autopilot [120], drug discovery in the pharmaceutical industry [157, 160], etc. However, despite its achievements, FL encounters several challenges in real-world environments, such as addressing training data class imbalance and enhancing robustness against vulnerabilities.

The class imbalance issue is one typical challenge that frequently happens in practical scenarios of FL [65, 147, 34]. Specifically, when a model encounters class imbalance, the classes with a large amount of training data (called majority class) can quickly reach a high testing accuracy while the accuracy of classes with less training data (called minority classes) remains low [159]. Furthermore, the client owning a large ratio of overall classes is usually more "promising" than the client owning less, which occupies a larger weight in the aggregation step, leading to biased training to the specific class [85].

Furthermore, the training data in real-world FL applications are usually Non-Independent and Identically Distributed (non-IID) [88, 84, 24]. Specifically, one client may only own a few particular classes of training data instead of all classes for the learning task[88]. For instance, [52] indicates some mammals (e.g., panda, kangaroo, etc.) only exist in particular demographic locations. In other words, a China mammal photo collection may not include the kangaroo and penguin. [26] shows students from different disciplines have very different library usage patterns; most students only search the research material of their disciplines instead of all subjects.

In FL, another major challenge is that FL is vulnerable to various adversarial manipulations due to its distributed nature [10, 60]. Specifically, the attacker can poison its training data owned and submit the corresponding gradient to corrupt the global model (known as data

poisoning attacks [8, 55]) or directly manipulate the local model updates (known as model poisoning attacks [154]). Based on the attacking objective, the attacks can be further classified as targeted and untargeted attacks, which aim to degrade the overall accuracy of the global model [37] or lead the model to misbehave on the targeted sub-tasks [42]. To address the malicious attacks, recent research introduces the byzantine-robust aggregation rule on the server to replace the averaging method for aggregation [9, 167]. One popular kind of the byzantine-robust aggregation rule (name “honest based weighted aggregation") prepares a small dataset on the server side and utilizes it to evaluate the clients’ honesty and determine the clients’ importance in aggregation [16, 110]. By leveraging the dataset, the server can remove the statistical outliers and limit potential attackers’ influence when updating the global model. However, most of these studies have not considered FL’s class imbalance nature, leaving opportunities for poisoning attacks.

In this section, we first design and propose a new attack called Class Imbalance Attack for the byzantine-robust FL that relies on “honest based weighted aggregation.” Unlike the existing poisoning attack that degrades the overall performance of the global model, our Class Imbalance Attack aims to degrade the model’s testing accuracy on the targeted class(es). The Class Imbalance Attack is performed from two perspectives. On the one hand, the attacker monitors the global accuracy in several iterations to find the most vulnerable class (usually the minority class) as the targeted class to further worsen its learning performance. On the other hand, the attacker evenly collects the dataset from all classes but only excludes the targeted class(es). As the model generated by the Class Imbalance Attack carries more class information than the model of the normal client, the crafted model can continuously gain a heavyweight in aggregation and achieve better attack performance. Our results show the Class Imbalance Attack can effectively degrade the accuracy of the targeted class and subsequently decrease the global model’s overall accuracy under the state-of-the-art FL methods [16, 110].

We further propose a new Class-Balanced FL method with a contribution-wise byzantine-robust aggregation rule to enhance the robustness of FL. In our FL method, we follow the mainstream byzantine-robust FL setting to keep a small dataset in the server and assign an honest score (HS) to each gradient received. To avoid driving the minority class to a

disadvantage, we further introduce the contribution score ($ConS$) to each model received, where the $ConS$ is high if the model shows good performance on the disadvantaged class of the current global model. Finally, the server combines the HS and $ConS$ as the final score (FS) and computes the average of normalized gradients weighted by their FS as the global model update. Instead of using the traditional normalization method [16], we introduce a novel trimmed-normalization method to maximize the learning performance and defend against scaling attacks. We evaluate our Class-Balanced FL on MNIST [31], Fashion MNIST [152] with different data distributions and provide a case study on cifar-10 [75] with extreme setting. Our results show that the Class-Balanced FL is robust against different poisoning attacks [99, 37, 154], including our Class Imbalance Attack.

4.2.2 Related Work

Class imbalance and robustness vulnerability are two critical FL issues that have been widely but separately studied by recent research [147, 126, 44, 58].

On the one hand, the current studies consider the class imbalance in non-adversarial FL settings. [147] introduces a minoring scheme and designs a new loss function (Ratio Loss) to mitigate the impact of the imbalance. In each iteration, the monitor feeds auxiliary data to the last global model and then generates the gradient updates on each class. By comparing the updates with the current global model, the system can detect the class imbalance composition and mitigate the impact by applying the Ratio Loss. [126] proposes a novel agnostic constrained learning formulation to tackle the class imbalance problem. With the introduced constraints, the classifier is forced to evenly account for all classes and hence moderate the class imbalance's effect.

On the other hand, most existing FL methods enhance the robustness of FL by introducing byzantine-robust aggregation rules under Class-Balanced training data [16, 131, 167]. Expect the Krum, Median, and Trimmed-Mean methods introduced in Section 3.2, we note that some recent research proposes a new kind of FL method that addresses the byzantine failure of federated learning by evaluating the clients' honesty based on a small dataset prepared.

FLTrust [16] leverages the small dataset (called root dataset) to generate and maintain a server version model. In each iteration, FLTrust first assigns a trust score to each gradient received based on the cosine similarity between the gradients of clients and the server. Then, the server normalizes the client gradients by scaling them to have the same magnitude as the server gradient. Finally, the server averages the normalized gradients weighted by the trust score and updates the global model.

Most recently, SageFlow [110] is proposed as a state-of-the-art two-stage aggregation rule, which includes entropy-based filtering and loss-weighted averaging. When SageFlow receives gradients from devices, the server measures the Shannon entropy of each gradient through the small dataset and consequently discards the gradient receiving a high entropy value. After filtering, SageFlow also measures the loss of each client's model and finally averages the gradients based on their loss values. The gradient achieving a low loss value may receive large importance in the aggregation and vice versa.

In this study, our primary focus lies on FL methods [16, 110] that rely on a pristine evaluation dataset (or harness the dataset to generate root gradients). We identify their limitations and propose novel solutions to further improve their performance.

4.2.3 Motivation

4.2.3.1 Problem Setup

Our problem is formulated on a multi-class classifier. We use $x \in \mathcal{X}$ to denote the feature space and let $y \in \mathcal{Y} = \{1, ..., C\}$ be the label space, where C is the total number of classes. We assume n clients participate in the learning task via training the local model $w_i, i = 1, 2, \ldots, n$ on their private data (x, y), and the local model updates g_i will be submitted to learn a shared global model G. g_i is the difference between its local model w_i and the optimal local model w_i^*, where w_i^* is a solution of client i for the following optimization problem:

$$w_i^* = argmin_{w_i} F(w_i) \tag{4.25}$$

Here, $F(g_i) = \mathbb{E}_{(x,y)\sim p_i}[l((x, w_i), y)]$ is the expectation of the empirical loss $l((x, g_i), y)$ and $p_i(x, y)$ is an unknown joint distribution of feature x and label y. $p_i(x, y)$ can be further composed as following,

$$p_i(x, y) = p(x|y)p_i(y) \tag{4.26}$$

where $p(x|y)$ denotes the conditional distribution of feature x given label y and $p_i(y)$ is the marginal distribution of label y on client i. We note that the conditional distribution $p(x|y)$ could be similar on all clients while the marginal distribution $p_i(y)$ can be significantly different in the FL context. Specifically, because of the heterogeneity of clients, client i may own a large number of data with label c but little (even 0) data with label $q, q \neq c$.

In the standard vanilla FL [97], the server updates the global model G^t by averaging the local model updates received, where t denotes the communication round, and η is the learning rate:

$$G^{t+1} = G^t + \eta \frac{1}{n} \sum_{i=1}^{n} g_i^t \tag{4.27}$$

Although vanilla FL can effectively aggregate the local models in a non-attackers setting, the averaging-based aggregation rule has been proven neither roust nor tolerate a single byzantine worker [9].

To mitigate the research gap, recent researches [16, 110] replace the aggregation rule from (4.27) to (4.28), where β_i denotes the weight of the i-th local model update.

$$G_{t+1} = G^t + \eta \frac{1}{n} \sum_{i=1}^{n} g_i^t \beta_i^t; \ \sum_{i=1}^{n} \beta_i^t = 1, \ \beta_i^t \in (0, 1) \tag{4.28}$$

Specifically, a small server-side dataset is retained for evaluating the client's updates. Based on the evaluation, the potential attackers and suspicious clients will be assigned a small weight in the aggregation and vice versa.

4.2.3.2 Class Imbalance Attack

In this paragraph, we introduce our Class Imbalance Attack. We focus on the case where the FL methods keep a dataset on the server side and aggregate local model updates through the aggregation rule (4.28). Our Class Imbalance Attack is designed based on two FL facts: (I)

Clients' local training data are non-IID, and the class imbalance issue exists. (II) Most clients only have the training data for a subset of the classes (i.e., for a benign client i, $\mathcal{Y}_i \subset \mathcal{Y}$, where $\mathcal{Y}_i$ is the label space of the training data).

In this work, we redefine the term "Adversary" to encompass both "intentional attackers(adversary)" and "unintentional attackers(adversary)." "intentional attackers" refer to participants who deliberately manipulate their owned data to create a specific data distribution, aiming to reduce the accuracy of targeted classes. On the other hand, "unintentional attackers" are clients who naturally own the specific data distribution but have no intention of compromising the global model.

We follow general assumptions [37, 8] that the current malicious participants can coordinate the attack to the current model update. Further, the attackers can prepare/own a training dataset that includes all classes and use this dataset to generate the malicious gradient.

Different from the existing data poisoning attacks [8, 104] that generate the perturbed gradients through craft malicious training data and degrade the model's overall accuracy, our strategy is generating the malicious gradient $\tilde{g}$ with a particular distribution of the benign training data ($\tilde{x}, \tilde{y}$; where $\tilde{y} \in \tilde{\mathcal{Y}}$ and $\mathcal{Y}_i \subset \tilde{\mathcal{Y}} \subset \mathcal{Y}$), submitting $\tilde{g}$ in each iteration, and finally disabling the global model G to identify the target class C, where $\mathcal{Y} - \tilde{\mathcal{Y}} = C$. Two key points for achieving the attack objective (i.e., disabling the global model G to identify the target class C) are Selecting the effective target class C for the attack and Guaranteeing the malicious gradient $\tilde{g}$ can obtain a heavyweight β in aggregation.

Because of the class imbalance issue in FL, the class that owns less global training data is more vulnerable and usually suffers from a slow learning speed. Thus, our strategy for the first key point is to attack the minority class C, which has the lowest testing accuracy from the global model G. To address the second challenge, we keep our malicious label space $\tilde{\mathcal{Y}}$ tightly close to the whole space and guarantee the malicious models perform better (e.g., a higher cosine similarity, a lower loss value, etc.) than the most benign models. Specifically, the Class Imbalance Attack is performed by the following steps:

Step I: Attacking Target Class Selection

Before performing the attack, the attackers monitor the global model G in several interactions (first 5%) and evaluate the accuracy of G on all classes. Suppose G achieves the lowest accuracy on label C; attackers determine the class C as the attack target. (We note that the targeted class C can further extend to two classes, and we demonstrate it in the following subsection.)

Step II: Crafted Training Dataset Preparation

The attackers build the crafted training data once the target class C has been determined. The training data should evenly include high-quantity/quality data from all classes except class C.

Step III: Class Imbalance Attack Performing

In each iteration, the attackers normally train the current global model G on our crafted data and send the malicious gradient $\tilde{g}$ back to the server.

The attackers perform steps I and II only once and keep the attack target (i.e., class C) constant during the whole attack.

The objective of gathering a substantial amount of high-quality data for adversarial training (mentioned in Step II) is to ensure its similarity to the evaluation dataset owned by the server and guarantee the generated malicious gradient can be evaluated as the most honest. Specifically, as the evaluation dataset of the server is manually collected from the real-world data resource[16], a large volume collection can ensure the adversary achieves a similar data distribution to the server. On the other hand, the high "quality" indicates the training data collected should be high resolution and representative to avoid label skew [116] between malicious gradients generating and gradients evaluation by the server.

4.2.3.3 Effectiveness of the Class Imbalance Attack

We evaluate our Class Imbalance Attack on two state-of-the-art byzantine-robust aggregation rules (4.28): FLTrust [16] and SageFlow [110] when adversarial clients occupy 5%, 15%, and 25% overall participants. The result shows our malicious gradient can continuously gain a

large weight in aggregation and effectively degrade the global model's accuracy to 0 on the target class when 25% participants are malicious (intentional or unintentional).

Experiment Setup

We assume 20 clients participate in the training task and evaluate the effectiveness of our Class Imbalance Attack when the adversary makes up 5%, 15%, and 25% overall participants. To simulate and reflect the class imbalance in FL, we introduce the parameter Γ by defining it as the ratio between the total sample number of majority class and minority class across all clients, i.e., $\Gamma = max_c(\sum_i N_p^i)/min_c(\sum_i N_p^i)$, where N_p^i denotes the number of samples for class c on client i. We introduce a scenario in which each client owns 2 or 3 classes of data and further extend the scenario to more general, that each client owns 1 to 4 classes. Specifically, we choose MNIST datasets with Γ as around 2: MNIST(2$\sim$3) and MNIST(1$\sim$4) to simulate the data distribution in the real world, where each client has 2 or 3 classes of data (i.e., $n(\mathcal{Y}_i) = 2, 3$), and has 1, 2, 3 or 4 classes of data (i.e., $n(\mathcal{Y}_i) = 1, 2, 3, 4$), respectively. The MNIST (Modified National Institute of Standards and Technology [31]) database is an extensive database of handwritten digits that includes 60,000 training images and 10,000 testing images; each image has consisted of 28×28 pixels and a label of $0 \sim 9$ to stand for the class belonging to.

Figure 4.10 illustrates the data distribution of benign clients for MNIST(2$\sim$ 3) and MNIST(1$\sim$4); once the client has been corrupted, the malicious can use the shared dataset/or collect more data to generate the adversarial gradient. In the learning task, client 0 acts as the server and owns the full label data set on the server side, client 1 to client 5 play the attacker's role based on different adversary volumes, and all other clients are normal participants.

As FLTrust and SageFlow require collecting a small clean dataset on the server side, we assume the server owns 5% (i.e., 500) of overall training data, and the dataset evenly covers all classes. We use a general model for training the MNIST classifier model. This model consists of a dense layer (28 * 28) and a softmax layer (10). We set the learning rate as 0.01, the batch size as 128, and the epoch as 50 to train the global model for 100 iterations.

Since the Class Imbalance Attack aims to degrade the model's accuracy on a target label and degrade its global accuracy subsequently, we monitor the global accuracy and the testing accuracy on the target label to evaluate the effectiveness of our attack. The testing accuracy reflects the model's robustness against byzantine attacks; in other words, it is more robust if the model has a higher testing accuracy. As our Class Imbalance Attack is performed to gain heavy weights, FedAvg is naturally robust to the Class Imbalance Attack, and we use it as the baseline to compare against.

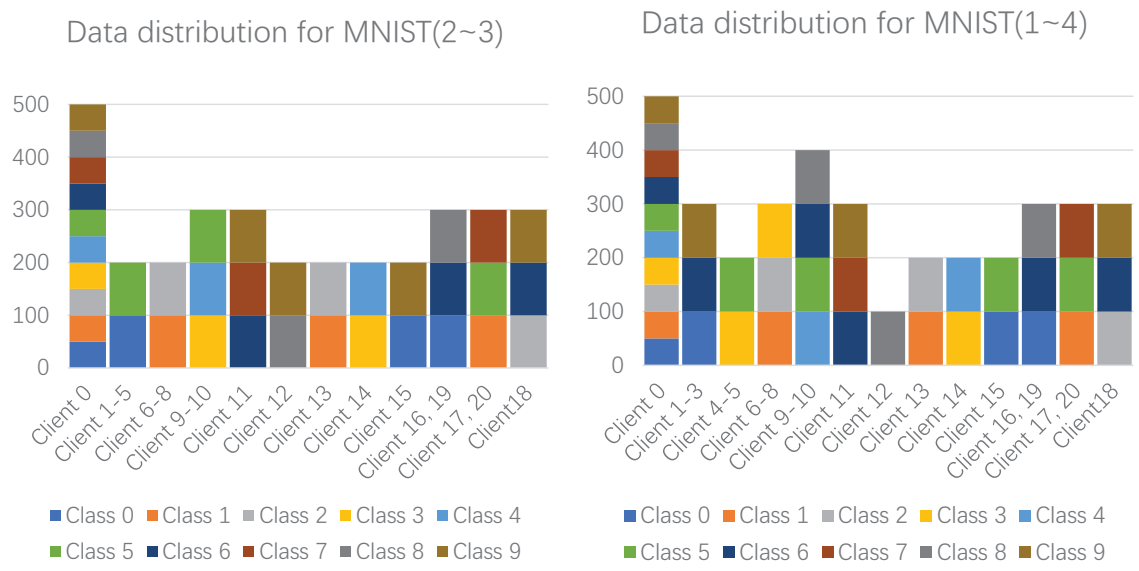

FIGURE 4.10: Illustration of the data distribution.

Experimental Results

The experimental results reveal the following findings: (I) In the proposed environment setting, the Class Imbalance Attack demonstrates its effectiveness when the adversary comprises 15% of the overall participants and reduces the testing accuracy of the targeted class to 0%, reaching the best-attacking performance when the proportion of malicious participants reaches 25%. (II) The FLTrust algorithm achieves a lower testing accuracy in non-IID scenarios, even in the absence of adversaries.

In the scenario where the adversary targets two specific labels and comprises 5% of the participants' capacity, existing FL methods are able to preserve the testing accuracy on the attacked class (Class 0 and 6). We note that FLTrust exhibits a lower global accuracy (around

73%) compared to FedAvg (83% and 78%) and SageFlow (around 84%) in MNIST(2$\sim$ 3) and MNIST(1$\sim$ 4) datasets, respectively. The attack demonstrates its effectiveness when the adversary reaches an occupation rate of 15%. Specifically, FLTrust reduces the accuracy to 20% on Class 6 in the MNIST(2$\sim$ 3) dataset, while SageFlow attains 0% accuracy on Class 6 in both the MNIST(2$\sim$ 3) and MNIST(2$\sim$ 3) datasets. In contrast, the baseline method maintains an accuracy of around 90%. Our Class Imbalance Attack demonstrates its highest effectiveness when the adversary comprises 25% of the participants. In such a scenario, both FLTrust and SageFlow achieve 0% accuracy on the attacked class, resulting in a significant drop in global accuracy. Specifically, FLTrust and SageFlow experience a decrease to around 65% and 68% accuracy on the respective datasets. Figure 4.11 illustrates the global accuracy of the existing FL methods against different amounts of adversaries. Figure 4.12 illustrates the attacked class accuracy of the existing FL methods against different amounts of adversaries.

We further evaluate the effectiveness of our Class Imbalance Attack when setting the attack strategy as attacking only one label in the scenario that 25% participants are adversaries. The experimental results show our attack achieves a significant performance; all attacked classes receive a 0% testing accuracy. As a result, FL Trust and SageFlow attain global accuracies of approximately 74% and 76%, respectively, when targeting Class 0. Similarly, when targeting Class 6, their global accuracies reach approximately 75% and 77%, which are lower than the baseline. Figure 4.13 illustrates the global accuracy of the existing FL methods when defending against Class Imbalanced Attack. Although Class 6 is more vulnerable due to occupying less volume capacity in our proposed experiments, it is noteworthy that attacking Class 0 can still result in achieving a 0% testing accuracy. In other words, the Class Imbalance Attack can effectively degrade the class learning performance even if it is not the most vulnerable[3] while attacking the most vulnerable class can achieve the best-attacking performance.

We note that, although FedAvg is naturally robust to our Class Imbalance Attack as it distributes a fixed weight to each client in aggregation, it suffers a slower learning speed when

[3]The vulnerable class we mentioned here is the class that achieves a low accuracy.

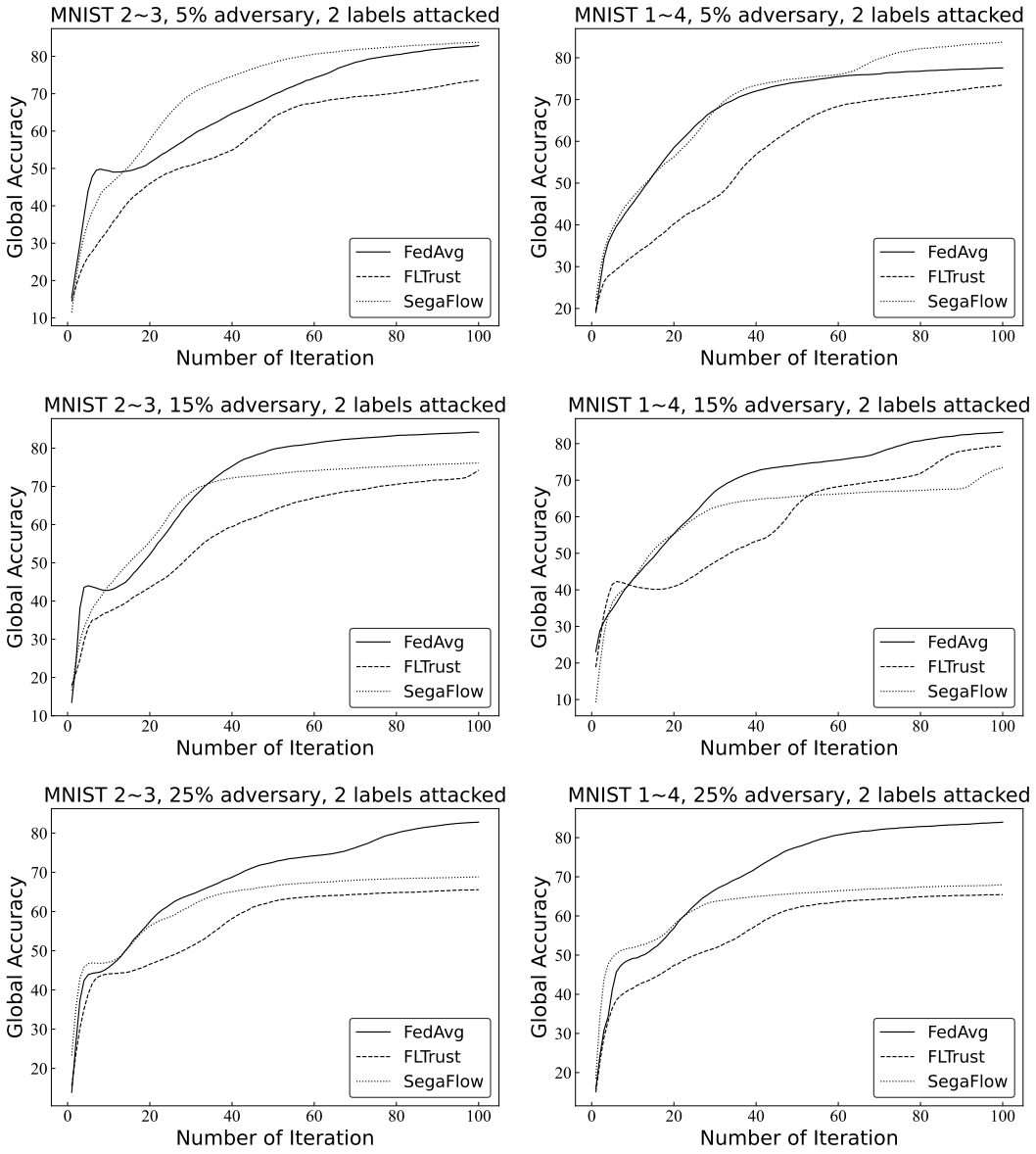

FIGURE 4.11: Global accuracy of the FLTrust, SageFlow, and FedAvg method against Class Imbalance Attack with dataset MNIST(2$\sim$ 3) and MNIST(1$\sim$ 4), where Class 0 and 6 are both attacked.

comparing the existing FL methods and can not achieve the best learning performance on the attacked class, we provide the related discussion in Section 4.2.5.

As the Class Imbalance Attack leverages the heavy weight gained in aggregation to achieve the attack objective, we monitor the weight distribution for each attacker under FL Trust, SageFlow, and Fedavg under Class Imbalance Attack when Class 0 has been attacked. The

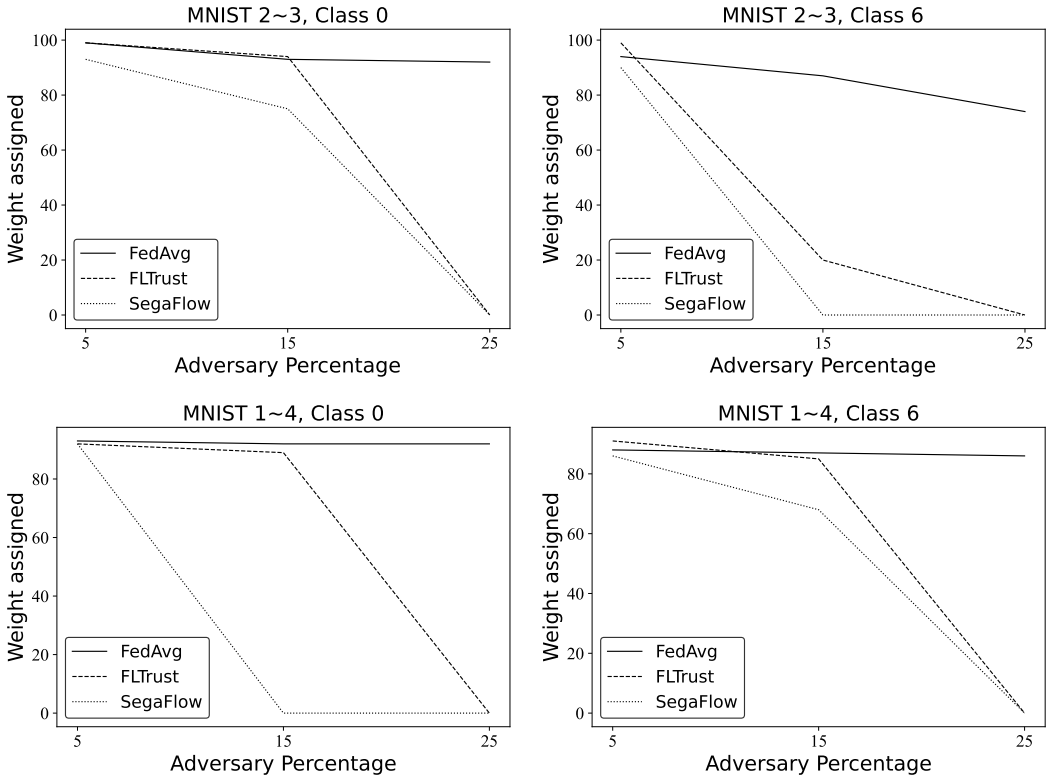

FIGURE 4.12: The class accuracy of the FLTrust, SageFlow, and FedAvg method against Class Imbalance Attack, Class 0 and 6 are both attacked.

monitor result shows the attackers can gain around 2 times more weight than other clients on average, which demonstrates the effectiveness of our attack. Table 4.8 illustration of the average weight gained per attacker in FLTrust/SageFlow and client in FedAvg.

TABLE 4.8: Illustration of the average weight gained by each attacker in FLTrust, SageFlow, and normal client in FedAvg.

FL Methods	FLTrust	SageFlow	FedAvg (normal client)
MNIST(2∼3)	10.62%	10.90%	5%
MNIST(1∼4)	10.72%	10.89%	5%

4.2.3.4 Discussion

Now, we discuss why our Class Imbalance Attack can effectively degrade the accuracy of the target label against FLTrust [16] and SageFlow [110].

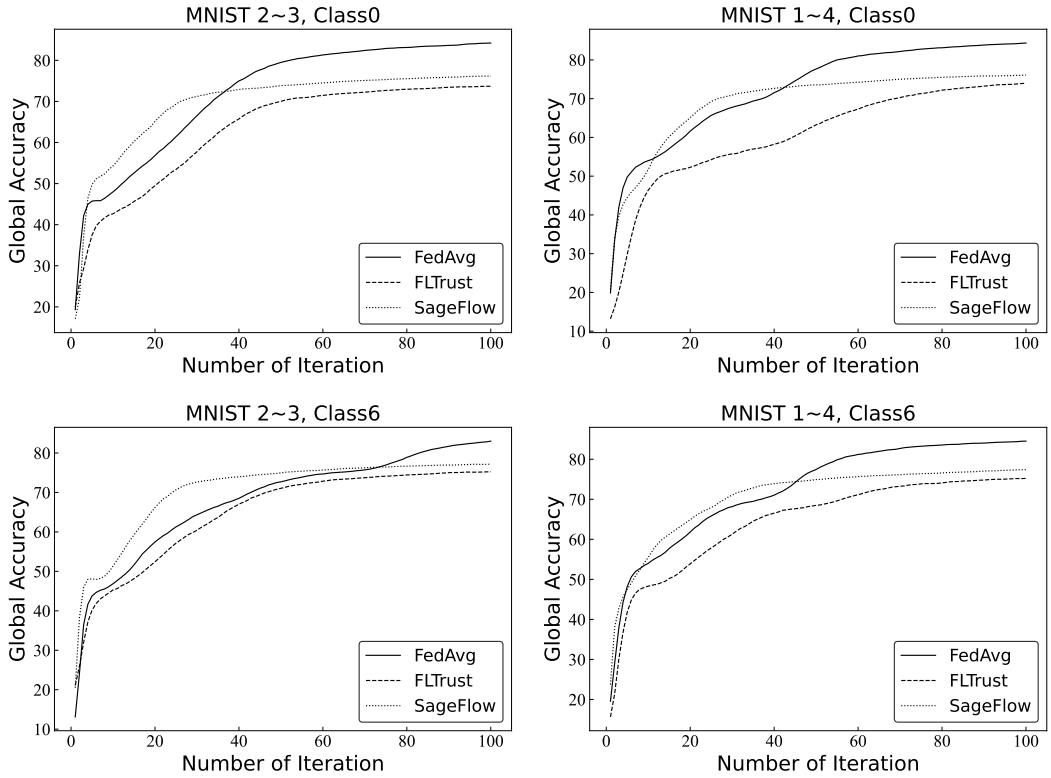

FIGURE 4.13: The global accuracy of the FLTrust, SageFlow, and FedAvg method against Class Imbalance Attack, one class is attacked.

Attacking FLTrust

Recall that FLTrust [16] keeps a small dataset (called root dataset) on the server side and maintains a server model using the root dataset. In each iteration, FLTrust calculates the cosine similarity of the gradients between the clients and server model; the gradient with a higher similarity will be assigned heavy importance (i.e., β) in the aggregation.

However, we find that in our proposed setting, not only the suspicious but also the clients with fewer classes gain a small weight in the FLTrust. Specifically, when the server gradient g_0 is generated on full labels, the gradient that owns fewer classes tends to have lower cosine similarity with g_0 and vice versa:

$$CS\left\langle g_i, g_0\right\rangle > CS\left\langle g_j, g_0\right\rangle, \text{ if } n(\mathcal{Y}_i) >> n(\mathcal{Y}_j) \tag{4.29}$$

Table 4.9 compares the cosine similarity and weight that g_i gained, where g_i owns the different number of classes data under the MNIST(1$\sim$4) setting. The gradient that owns 9 classes

receives the highest cosine similarity and consequently gains the heaviest weight: $\beta_i = 10\%$. In contrast, the gradient with 1 class receives less than 0.05 cosine similarity and little importance($< 2\%$) in aggregation.

TABLE 4.9: Illustration of the cosine similarity and corresponding weight gained by different g_i, under MNIST(1∼4).

Gradient	g_i, $n(\mathcal{Y}_i) = 9$	g_i, $n(\mathcal{Y}_i) = 2 \sim 4$	g_i, $n(\mathcal{Y}_i) = 1$
Cos similarity	> 0.70	$0.22 \sim 0.43$	< 0.05
Weight gained	$> 9\%$	$3\% \sim 5\%$	$< 2\%$

The Class Imbalance Attack leverages and further worsens the issue of FLTrust by crafting and submitting the "giant" gradient (only excludes the minority class.) Like g_i, $n(\mathcal{Y}_i) = 9$, the crafted gradients can keep a high cosine similarity and gain huge importance in aggregation. The global model can normally learn the classes owned by the malicious gradient. On the contrary, the class excluded can hardly be learned due to the insufficient weight gained and finally leads the global model to receive 0% accuracy on the label attacked.

Furthermore, we note that the FL Trust achieves a lower global accuracy even if the adversary has not performed the Class Imbalance Attack successfully (refer to the learning pattern when 5% adversaries participate in learning tasks shown in Fig 4.11, 4.12 which demonstrates the global accuracy of different in non-IID settings). Recall FL Trust leverages a fixed normalizing gradient aggregation that normalizes all client gradients based on the server(root) gradient. The intuition is that fixed normalizing limits the gradient magnitudes of not only adversaries but also benign clients. We consequently monitor the normalizing process in MNIST datasets with 5% adversarial participants; we find that although no scaling attack is performed, almost all magnitudes of clients' gradients have been normalized as half ($0.39 \sim 0.62$) in the aggregation. The observation illustrates using the server gradient (with all classes of training data) to normalize the client gradient (without all classes) is not proper in non-IID scenarios, which may result in degraded learning performance.

Attacking SageFlow

SageFlow [110] collects a small amount of validation data on the server side and evaluates the

gradients received from "entropy"(called entropy-based filtering) and "loss"(loss-weighted averaging).

Entropy-based filtering: When the server receives the gradients from clients, SageFlow measures their Shannon entropy through the validation data and discards the gradients having higher entropy values (around 2.5). However, we find that except for the malicious gradients, the gradient that is benign but owns little class also receives a high entropy value and is discarded in SageFlow. Table 4.10 compares the entropy of gradients owning different numbers of classes and from null attack under MNIST(1∼4). As the malicious gradients crafted by the Class Imbalance Attack own most classes like g_i, $n(\mathcal{Y}_i) = 8$, SageFlow can not filter out these malicious gradients by comparing entropy values. In contrast, the gradients from benign clients owing little class of data(e.g., g_i, $n(\mathcal{Y}_i) = 2$) achieve a high entropy value and are further discarded by SageFlow.

TABLE 4.10: Illustration of the Shannon entropy of different g_i in MNIST(1∼4).

Gradient	g_i, $n(\mathcal{Y}_i) = 10$	g_i, $n(\mathcal{Y}_i) = 8$	g_i, $n(\mathcal{Y}_i) = 2$	g_i, under null attack
Entropy	0.48	0.49	2.48	2.53

Loss-weighted averaging: SageFlow further measures the loss of each gradient received through the validation dataset. Based on the loss value, SageFlow assigns the weight by giving importance to the gradient, achieving a low loss value on the validation dataset and vice versa. Like "entropy-based filtering," the attackers can constantly receive a high weight through owning most classes and further expels clients owning fewer classes to participate in aggregation.

Based on the discussion, the disadvantages of the existing honest-evaluation-based FL methods (e.g., FLTrust, SageFlow) can be summarized as follows: (I) Lacking dynamic regulation mechanism of weight β: As the gradient's honesty (reflects in cosine similarity and loss value) highly depends on the data quality and distribution, the weight allocation β is relatively inflexible," under these FL methods unless clients significantly change their dataset. (II) Lacking importance on single label accuracy: As FLTrust and SageFlow only focus on the

full label (overall) accuracy, the client with high testing accuracy on a single label (C) can hardly participate in the aggregation.

4.2.4 Class-Balanced Federated Learning

4.2.4.1 Overview of Class-Balanced FL

Motivated by the Class Imbalance Attack, we propose a new FL method named Class-Balanced Federated Learning, which relies on the Contribution-Wise byzantine-robust aggregation rule to achieve byzantine robustness against malicious attacks.

The objectives of Class-Balanced FL are shown as follows:

I. Maintain the overall global accuracy in non-malicious scenarios: The Class-Balanced FL should achieve a similar testing accuracy to other FL methods when there is no attack.
II. Robust against the existing FL model poisoning attacks: The Class-Balanced FL should maintain the model's global accuracy when defending against the existing FL poisoning attacks.
III. Robust against the Class Imbalance Attacks: The Class-Balanced FL should maintain the model's global accuracy when defending against our Class Imbalance Attacks.

Different from the existing FL methods that only evaluate the clients' gradient from the "honest" perspective, our Class-Balance FL also considers the potential contribution of each local model update. In the Class-Balance FL, each client is assigned an Honest Score (HS) and a Contribution Score (CS) to decide its weight in the aggregation.

Specifically, the Class-Balanced FL keeps a small and clean dataset on the server side and first calculates the Honest score (HS) for each client's gradient. Owing to the excellent generalization, our Class-Balanced FL can utilize different honest evaluation algorithms, and we select two state-of-the-art evaluation algorithms (i.e., "Cosine similarity" and "Loss value" based algorithms) in this paper. Then, the server randomly selects the same amount of data in each class from the data owned as this iteration's validation data. The Class-Balanced FL evaluates the current global model through the validation data and generates an importance

vector (IV) based on its performance in each class. The class with low accuracy will achieve a high value in the importance vector. We further evaluate the performance of the gradient received on each class through the validation data to generate the client's performance vector (PV) and consequently generate each gradient's contribution score ($ConS$) by timing the global importance vector with their performance vector. The server finally combines HS and $ConS$ as FS (i.e., final score) and computes the average of the gradients weighted by FS as the global gradient, which is used to update the global model. Instead of using the original magnitude of the gradient in aggregation, we introduce a novel trimmed-normalized aggregation method to decide the gradient's magnitude, guaranteeing the learning performance and defending against scaling attacks.

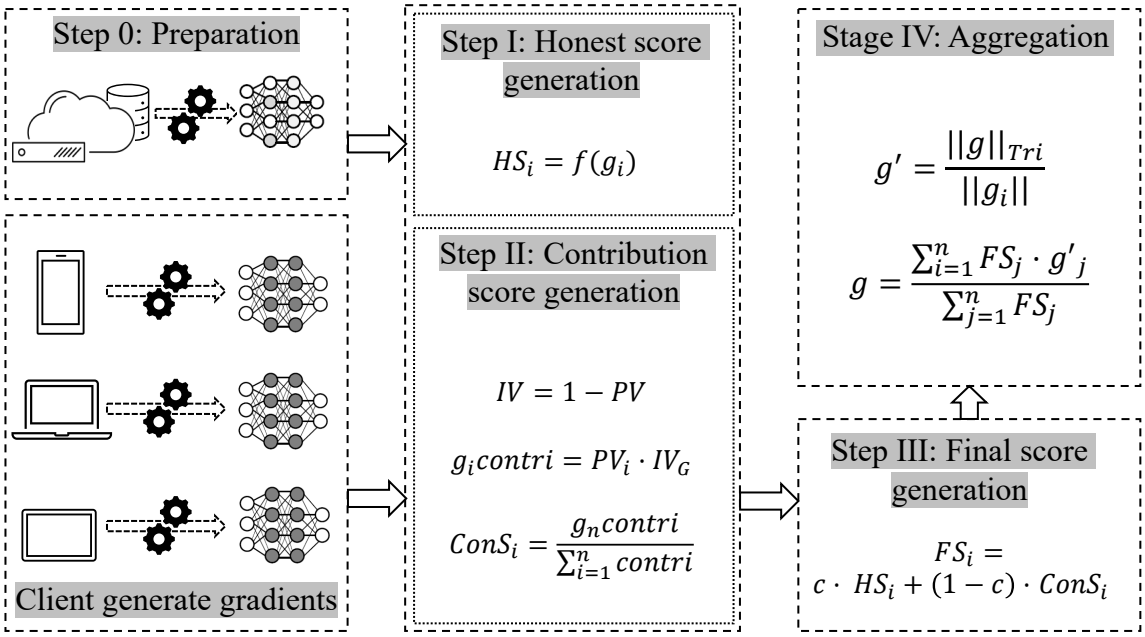

FIGURE 4.14: Contribution-Wise byzantine-robust aggregation rule.

4.2.4.2 Contribution-Wise Byzantine-Robust Aggregation Rule

The Contribution-Wise byzantine-robust Aggregation Rule aims to enhance the dynamics of FL by considering two perspectives: weight distribution and magnitude participation. Firstly, in terms of weight distribution, the aggregation rule should dynamically adjust the weights assigned to each participant based on the current global model, ensuring the learning

performance of both the minority class(es) and the majority class(es) is taken into account. On the other hand, the aggregation rule should dynamically determine the gradient norm of each client during the aggregation process to ensure a suitable learning step and defend against scaling attacks. Figure 4.14 illustrates the Contribution-Wise byzantine-Robust aggregation rule.

Note that the Class-Balanced FL leverages three different versions of gradients in the training process: the clients' gradients generated by each client through the local data, the server gradient generated by the server through the small clean dataset owned, the global gradient generated by the server through Contribution-Wise byzantine-Robust aggregating the clients' gradients.

Step 0: Preparation

The server collects a small and clean dataset that covers all classes of the learning task. In the same way as the client training its local model, the server also generates the server gradient before aggregation. Furthermore, a preset amount of clients have been selected through a sampling method S to participate in the learning task and aggregation. The Contribution-Wise Byzantine-Robust aggregation could leverage any existing client sampling methods to meet various service requirements (e.g., fairness, bandwidth, etc.) In this proposed example, we specify the method S as random sampling.

Step I: Honest score generation

To guarantee the benign of each client, we first evaluate the local model updates' honesty through an honest evaluation function $f(\cdot)$ and then assign each client an Honest Score (HS). In this paper, we introduce two algorithms as the honest evaluation function (i.e., $f(\cdot)$), but our proposed Contribution-Wise byzantine Aggregation enjoys good generalization, which can utilize any honest evaluation method.

Under the "Cosine similarity" based algorithm [16], the server gradient is used to generate the Honest Score (HS) and lead the global model updates in the right direction. The gradient that is close to the server gradient (g_0) is more trustable and should be assigned a high HS. Specifically, we measure the cosine similarity between the gradient of client and server

through the following:

$$CS_i = \frac{< g_i, g_0 >}{||g_i|| \cdot ||g_0||} \tag{4.30}$$

As the cosine similarity (CS) is negative when two directions are opposite, we further use the ReLU operation to eliminate the negative impact and generate the Honest Score for each client. Formally, we have the following:

$$\begin{gathered} HS_i = ReLU(CS_i), \\ \text{where } ReLU(c) = c, \text{ if } c > 0 \text{ or } ReLU(c) = 0 \end{gathered} \tag{4.31}$$

Under the "Loss value" based algorithm [110], the server directly evaluates each client's model updates through the small dataset and generates the corresponding loss value. The client's Honest Score is inversely proportional to its loss value; formally, the HS is generated through the following:

$$HS_i \propto \frac{1}{L(g_i)}, \text{where} \sum_{i=1}^{n} HS_i = 1 \tag{4.32}$$

Here, $L(\cdot)$ is the loss function, and we use the cross-entropy loss in this paper.

Step II: Contribution score generation

As cosine similarity highly relies on data distribution, the clients who owe most classes can continuously receive a huge weight in FLTrust aggregation even if the global model has already learned their classes and received high accuracy. A malicious client can leverage this issue and leads the global model to achieve 0% accuracy on the label attacked by excluding only the target class in its training data. Therefore, we introduce the contribution score ($ConS$) to evaluate the potential contribution brought by the gradient.

In particular, the server first selects a part of the data owned as the validation data in this iteration; the validation dataset selected should contain all classes of the learning task. Then, the server evaluates the current global model through the validation dataset and generates the corresponding performance vector. The global performance vector (PV_G) illustrates the testing accuracy of the global model on different classes. Based on the PV_G, we generate the global importance vector (IV_G) to decide which class should be given more importance in

this iteration. Formally, the IV_G is defined as follows:

$$IV_G = 1 - PV_G \tag{4.33}$$

The server further evaluates the performance of each client's gradient through the same validation dataset and builds the associated performance vector ($PV_i, i = 1, 2, 3...n$); the contribution score of each client is finally generated by timing the IV_G by its PV_i. Formally, we have the following:

$$ConS_i = \frac{PV_i \cdot \ IV_G}{\sum_{j=1}^{n} ConS_j} \tag{4.34}$$

Step III: Final score generation

In our Class-Balanced FL, the gradient's final score consists of both HS_i and $ConS_i$:

$$FS_i = c \cdot HS_i + (1 - c) \cdot ConS_i \tag{4.35}$$

where c is a hyper-parameter for balancing HS and $ConS$; (Based on the experiments, we have determined that setting $c = 0.4$ yields the optimal performance for the proposed FL framework. Therefore, without further notification, we will use $c = 0.4$ in this study.) By leveraging on the $ConS$, the server can dynamically distribute high weight to those gradients having potential high distribution. In contrast, the gradient may receive low importance when the class owned has been well learned by the global model, even if it achieves a high cosine similarity.

Step: IV: Aggregation

As directly introducing the client's gradient in aggregation enables the adversary to scale its gradient and dominate the global update, using a fixed normalized version gradient [16] may lag the learning performance; we follow the trimmed thought from [167] and propose a trimmed-normalized gradient (g') for aggregation.

Specifically, after generating the FS in Step III, the server calculates the $l2$ norm for each gradient and orders them by magnitude. The $p\%$ largest and smallest gradients will be discarded, and the largest and smallest magnitudes of the trimmed order will work as the

boundaries. In other words, in the current learning iteration, the gradients achieving over-trimmed boundaries will be normalized by the corresponding trimmed boundaries while others can keep their magnitude. Formally, we have the following:

$$\begin{aligned} g_i' &= \frac{||g||_{TriSmal}}{||g_i||} \cdot g_i, \text{ if } ||g_i|| < ||g||_{TriSmal} \\ \text{or } g_i' &= g_i, \text{ if } g_i \in [||g||_{TriSmal}, ..., ||g||_{TriLarg}] \\ \text{or } g_i' &= \frac{||g||_{TriLarg}}{||g_i||} \cdot g_i, \text{ if } ||g_i|| > ||g||_{TriLarg} \end{aligned} \tag{4.36}$$

We compute the global model update (i.e., g) by averaging the normalized gradient weighted by their FS:

$$g = \frac{1}{\sum_{j=1}^{n} FS_j} \sum_{i=1}^{n} FS_i \cdot g_i' \tag{4.37}$$

Finally, the server updates the global model through g to finish the t iteration:

$$G^{t+1} = G^t + \eta \cdot g \tag{4.38}$$

4.2.5 Evaluation

We evaluate our Class-Balanced FL (short as Class B FL) against the existing FL poisoning attacks [99, 154, 37] and the designed Class Imbalance Attack in this section. To demonstrate the generality of the Class-Balanced FL, we follow Section IV to introduce "Cosine similarity, (short as CS)" and "Loss value, (short as LV)" based algorithm to generate the HS and evaluate them respectively. As the Class Imbalance Attack reaches its best performance when 25% participants are malicious, we use the setting that 25% of overall clients are adversarial (could be intentional or unintentional) to evaluate the robustness of our Class-Balanced FL.

4.2.5.1 Experimental Setup

We follow the setting introduced in the section "Effectiveness of the Class Imbalance Attack" and further use Fashion MNIST($2\sim 3$) and Fashion MNIST($1\sim 4$) when performing the Class Imbalance Attack. The Fashion MNIST (Fashion-MNIST [152]) dataset includes 60,000

gray-scale images of 10 fashion categories and a test set of 10,000 images. We follow the data distribution introduced in section III to initialize the training dataset for all participants.

When performing the existing poisoning attacks, we assume $C_1, C_2, C_{11}, C_{12}, C_{20}$ as attackers and further assign C_3, C_4, C_5 rest classes to generate MNIST(2$\sim$3)$_{v2}$ and MNIST(1$\sim$4)$_{v2}$ and maintain the same global training data capacity. We use the following state-of-the-art poisoning attacks to evaluate the robustness of the Class-Balanced FL:

I. Fall of empires attack [154]: Fall of empires attack (short as Fall attack) uploads the crafted gradient where gradient = $-z * \mu$. Here, μ is the mean of gradients based on the client occupied, and z is the attack multiple; we follow the setting in [154] and set $z = 10$.
II. Local model poisoning attack [37]: Local model poisoning attack (short as Local attack) first infers the convergence direction of the gradients and then uploads the scaled, reverse gradient to poison the global model.
III. Partial drop attack [99]: Partial drop attack (short as Drop attack) masks the gradient parameter as 0 with probability p. As the parameter naturally carries a few 0 in our training tasks, we replace the mask 0 as -10 to enhance the attack strength and set $p = 0.8$ in experiments.

TABLE 4.11: Illustration of the global accuracy of different existing and proposed FL methods against various poisoning attacks.

FL Methods	**FL Trust**	**SageFlow**	**Class Balanced FL(CS)**	**Class Balanced FL(LV)**
MNIST(2$\sim$3)$_{v2}$				
Fall Attack	0.84	0.83	0.84	0.84
Local Attack	0.84	0.84	0.84	0.84
Drop Attack	0.84	0.84	0.84	0.83
MNIST(1$\sim$4)$_{v2}$				
Fall Attack	0.82	0.83	0.82	0.82
Local Attack	0.83	0.84	0.83	0.83
Drop Attack	0.82	0.84	0.82	0.83

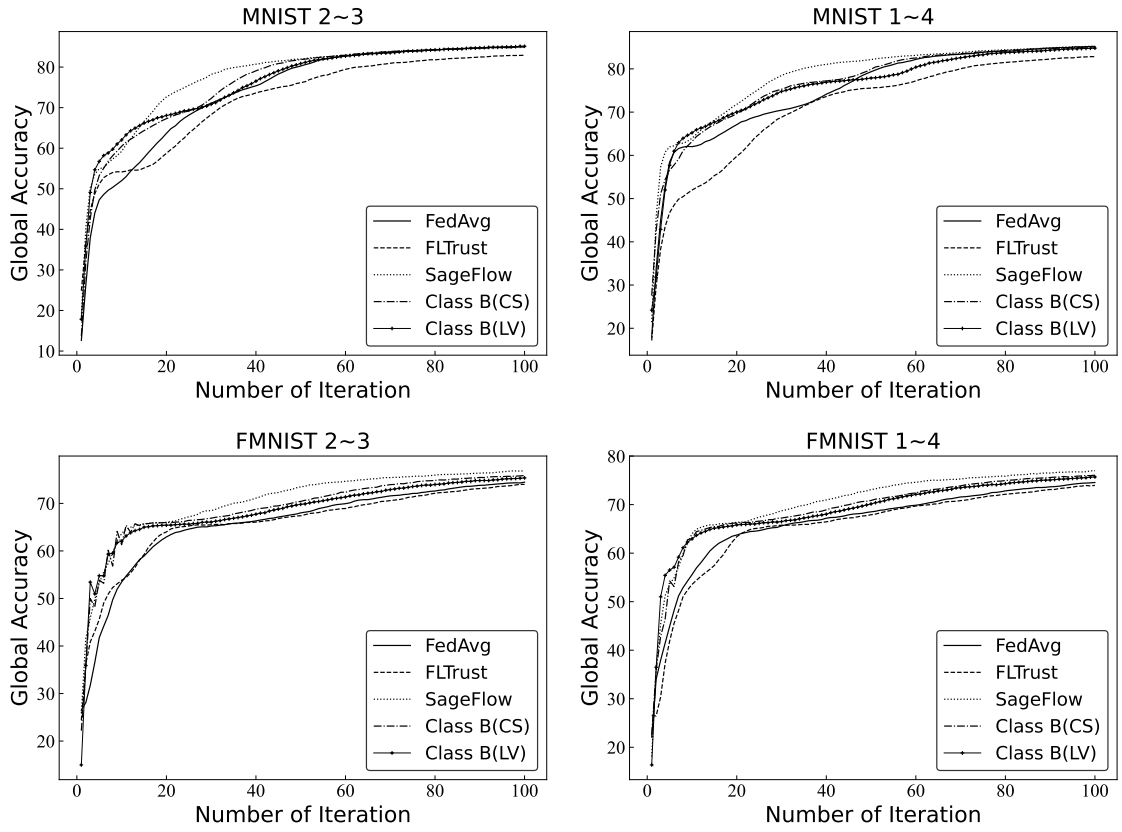

FIGURE 4.15: Illustration of the global accuracy of different FL methods in non-adversarial environments.

4.2.5.2 Experimental Results

The experimental results show that our Class-Balanced FL achieves all three objectives in Section 4.2.4.1. First, Our Class-Balanced FL achieves similar accuracy to the existing FL methods when there is no attack. For instance, the Class-Balanced FL(CS) and Class-Balanced FL(LV) receive the testing accuracy of 84.97% and 85.13% on non-adversarial MNIST(2$\sim$3) 85.07% and 84.79% on adversarial MNIST(1$\sim$4), while FLTrust achieves 82.91% and 82.94%, SageFlow achieves 85.01% and 85.22%, FedAvg achieves 85.11% and 85.19% on two datasets, respectively. We note that suffering from the improper normalizing magnitude of local updates (discussed in Section III, Part D), FLTrust achieves a lower testing accuracy on MNIST and Fashion MNIST serious datasets. Figure 4.15 illustrates the global accuracy of different FL methods in non-adversarial environments.

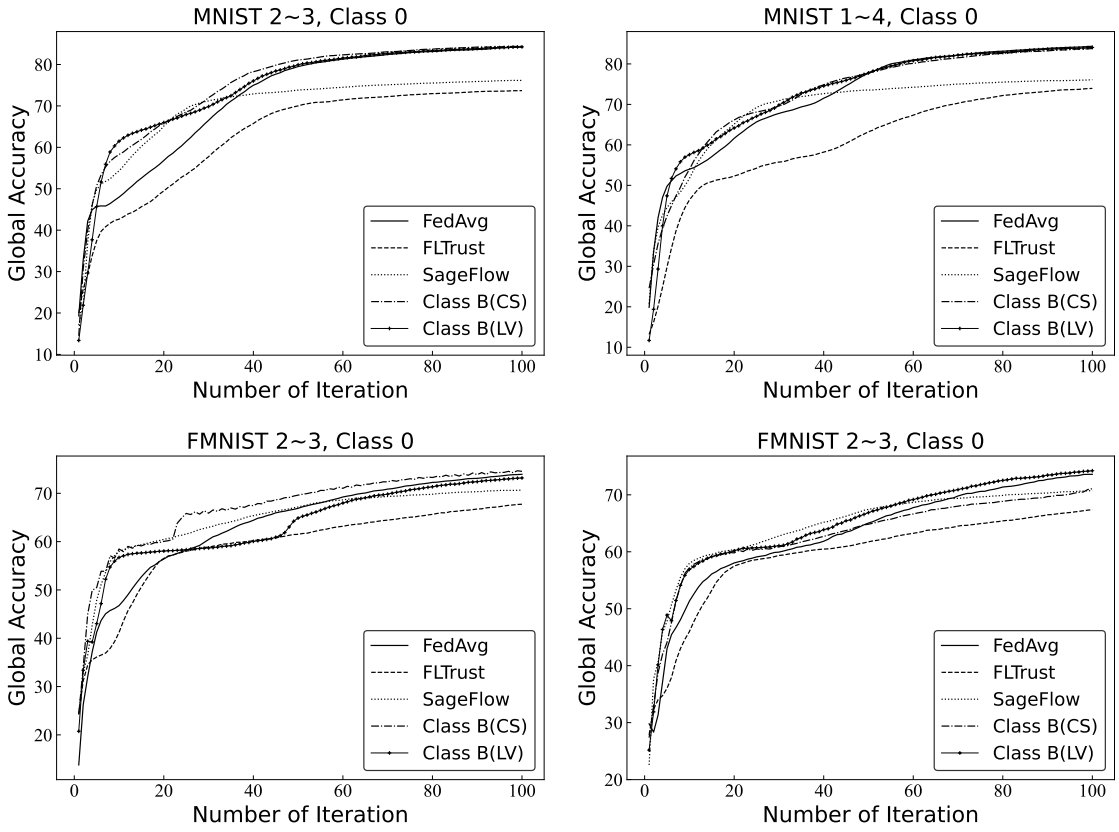

FIGURE 4.16: Illustration of the global accuracy of different FL methods against the Class Imbalance attack (Class 0 is attacked).

Second, the Class-Balanced FL is robust against the existing FL poisoning attacks. In particular, when defending against the "Fall," "Local," and "Drop" attacks, the Class-Balanced FL(CS) achieves 84.39%, 84.57%, and 84.23% testing accuracy on MNIST(2$\sim$3)$_{v2}$; achieves 82.41%, 83.22%, and 82.14% testing accuracy on MNIST(1$\sim$4)$_{v2}$ respectively. Similarly, the Class-Balanced FL(LV) achieves 84.21%, 84.04% and 83.46% testing accuracy on MNIST(2$\sim$3)$_{v2}$; achieves 82.29%, 83.84% and 83.46% testing accuracy on MNIST(1$\sim$4)$_{v2}$ respectively. Table 4.11 compares the global accuracy of various FL methods defending against different poisoning attacks.

Third, our Class-Balanced FL achieves satisfactory learning performance when defending against the Class Imbalance Attack. On the one hand, the Class-Balanced FL receives the highest testing accuracy on the attacked class in most cases. Specifically, in the MNIST serious datasets, Class-Balanced FL methods receive 80%+ accuracy on Class 6, and around

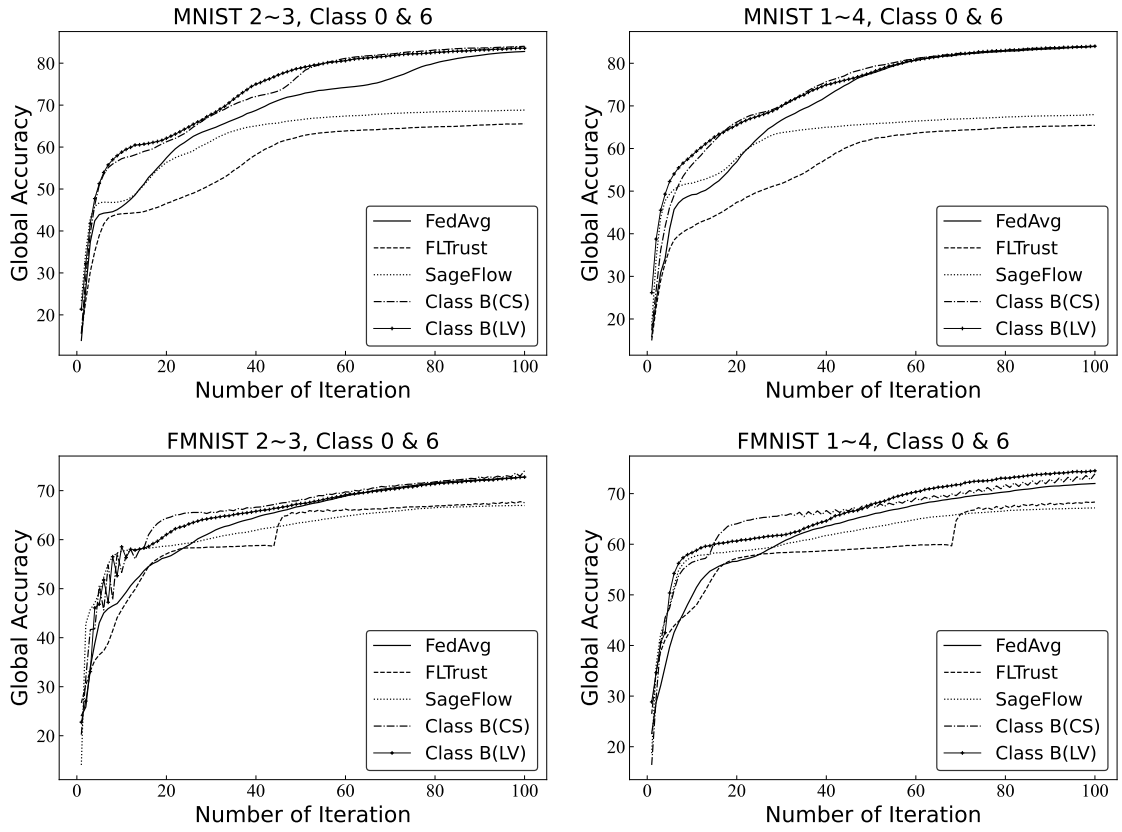

FIGURE 4.17: Illustration of the global accuracy of different FL methods against the Class Imbalance attack (Classes 0, 6 are attacked).

92% accuracy on Class 0 regardless of whether one or two classes are attacked. In contrast, SageFlow and FL Trust witness 0% accuracy on the attacked class in all MNIST-based scenarios. In FMNIST datasets, our Class-Balanced FL achieves around 78% accuracy on class 0, while the existing FL methods achieve 0%. Table 4.12 illustrates the accuracy of attacked classes under different FL methods; "✓" indicates the highest testing accuracy in its column.

On the other hand, the declining accuracy of the attacked class under the existing FL methods leads to a similar degrading trend in their overall global accuracy. For instance, due to one label being attacked, FL Trust and SageFlow decrease the global accuracy to around 73% and 76% in MNIST and 67% and 70% in FMNIST datasets, respectively. In contrast, Class-Balanced FLs can maintain a global accuracy of approximately 84% and 73.5% in two(serious) datasets.

TABLE 4.12: The accuracy of the attacked label when different FL methods against Class Imbalance Attacks.

Dataset	**MNIST(2∼3)**			**MNIST(1∼4)**		
Attacked Class	Only C 0	C 0	C 6	Only C 0	C 0	C 6
FedAvg	0.92	0.92	0.74	0.91	0.92	0.86
SageFlow	0.00	0.00	0.00	0.00	0.00	0.00
FL Trust	0.00	0.00	0.00	0.00	0.00	0.00
Class Balanced FL (CS)	0.93✓	0.93✓	0.87✓	0.93✓	0.93✓	0.88
Class Balanced FL (LV)	0.92	0.92	0.81	0.92	0.92	0.96✓
Dataset	**Fashion MNIST(2∼3)**			**Fashion MNIST(1∼4)**		
Attacked Class	Only C 0	C 0	C 6	Only C 0	C 0	C 6
FedAvg	0.71	0.79	0.00	0.59✓	0.78	0.03
SageFlow	0.00	0.00	0.00	0.00	0.00	0.00
FL Trust	0.00	0.83✓	0.00	0.00	0.81	0.00
Class Balanced FL (CS)	0.77	0.80	0.30✓	0.00	0.81✓	0.38
Class Balanced FL (LV)	0.78✓	0.79	0.00	0.57	0.57	0.57✓

By leveraging the proposed trimmed-normalized gradient aggregation method, our Class-Balanced FLs have witnessed a faster learning speed. For instance, under the non-adversarial setting, Class-Balanced FLs can reach 60% accuracy within 20 and 10 learning iterations in MNIST and FMNIST datasets, which is half of FL trust. This phenomenon also exists when adversaries are involved; for instance, although FL Trust achieves 83% accuracy in Class 0 of FMNSIT(2∼3), it achieves over 50% accuracy after the 46th learning rounds. In contrast, Class-Balanced FL (CS) and (LV) reach the same accuracy at the 16th and 25th iterations, respectively. We provide further discussion and comparison of with/without trimmed-normalized gradient aggregation method in the following discussion.

To further demonstrate the effectiveness of our proposed Class-Balanced FL, we extend our experiments to a border setting which introduce 100 clients and random sampling. Table 4.13 illustrates the extended experiments results. Align to the results shown before, our Class-Balanced FLs achieve the best learning performance and robustness in most cases.

TABLE 4.13: The accuracy of the attacked label when different FL methods against Class Imbalance Attacks (with 100 clients and random sampling).

Dataset	**MNIST(2∼3)**			**MNIST(1∼4)**		
Attacked Class	Only C 0	C 0	C 6	Only C 0	C 0	C 6
FedAvg	0.91	0.93	0.76	0.91	0.91	0.88
SageFlow	0.00	0.00	0.00	0.00	0.00	0.00
FL Trust	0.00	0.00	0.00	0.00	0.00	0.00
Class Balanced FL (CS)	0.92✓	0.94✓	0.87✓	0.92✓	0.90✓	0.87
Class Balanced FL (LV)	0.92✓	0.91	0.87✓	0.92✓	0.90	0.93✓
Dataset	**Fashion MNIST(2∼3)**			**Fashion MNIST(1∼4)**		
Attacked Class	Only C 0	C 0	C 6	Only C 0	C 0	C 6
FedAvg	0.70	0.76	0.08	0.56✓	0.78	0.06
SageFlow	0.00	0.00	0.00	0.00	0.00	0.00
FL Trust	0.00	0.80✓	0.00	0.00	0.82	0.00
Class Balanced FL (CS)	0.78✓	0.80✓	0.32✓	0.06	0.83✓	0.42
Class Balanced FL (LV)	0.76	0.80✓	0.00	0.56✓	0.58	0.55✓

We note that although FedAvg is naturally robust to the Class Imbalance Attack, it can not achieve the best learning performance in the attacked class. For example, FedAvg achieves 0% accuracy on class 6 under FMNIST(2∼3) while Class-Balanced FL (CS) can further increase the accuracy to 30%.

To demonstrate the effectiveness of our introduced Contribution Score (CS), we further conduct ablation experiments for algorithms under the Class Imbalance Attack scenario. As shown in Table 4.14, we monitor the global accuracy of the model during the Class Imbalance Attack (involving either one or two labels) using an aggregation rule that incorporates either only the Honest Score (HS) or both CS, HS. We observe that the algorithms, including FL Trust and SageFlow, were ineffective when solely employing HS for aggregation. The results of these experiments indicate that the integration of CS significantly enhances the model's robustness against Class Imbalance Attacks, leading to an accuracy improvement ranging from 6% to 19%.

TABLE 4.14: Illustration of the global accuracy of different existing and proposed FL methods against various Class Imbalance attacks. Progressing from left to right, it begins with FedAvg, then introduces the Imbalance Attack. Subsequently, the algorithms that rely on HS are depicted, followed by those employing both HS and CS.

Dataset	**FedAvg (Benign)**	**Imbalance Attack**	**FL Trust (With HS)**	**SageFlow (With HS)**	**Class B FL (With HS, CS)**
MNIST	0.83 $\rightarrow$	One Label	0.73 (-0.10)	0.76 (-0.07) $\rightarrow$	0.84 (+0.11)
FMNIST	0.73 $\rightarrow$	One Label	0.67 (-0.06)	0.70 (-0.03) $\rightarrow$	0.73 (+0.06)
MNIST	0.82 $\rightarrow$	Two Labels	0.68 (-0.14)	0.64 (-0.18) $\rightarrow$	0.83 (+0.19)
FMNIST	0.73 $\rightarrow$	Two Labels	0.65 (-0.08)	0.66 (-0.07) $\rightarrow$	0.72 (+0.07)

4.2.5.3 Case Study

To evaluate the effectiveness of our proposed Class-Balanced FL on large machine learning models, complicated learning tasks, and extreme settings. We further provide a case study on the cifar-10 [75] dataset and demonstrate the Class-Balanced FL achieves a better performance than the existing FL methods.

Case Study Setup

We consider that 5 clients have been selected to participate in the learning task, and 1 client is adversarial. We consider each normal client evenly owns 4 different classes of data, but ONLY one client owns data of Class 9; in other words, Class 9 is the minor class in this learning task. We followed the adversary set in Section III, which could be intentional or unintentional, selecting one class to attack (Class 9 in the practical experiments). We use cifar-10 as the training data; cifar-10 is a popular computer vision dataset for object recognition. It consists of 60,000 32x32 color images, divided into 10 classes with 6,000 images per class. We use a simplified version of ResNet [50] as the training model, which consists of two convolution groups (MaxPooling + CONV2D) and several flatten, dense layers. We train the model 100 iterations and compare our Class-Balanced FL (based on cosine similarity) with Fedavg, FL Trust, and SageFlow and use the accuracy (accuracy of global and minor class) as the metrics.

Case Study Results

The experimental results show our Class-Balanced FL achieves a better learning performance compared with the existing state-of-the-art FL methods.

From the global perspective, our Class-Balanced FL achieves a higher testing accuracy of around 48%, while SageFlow, FL Trust, and Fedavg receive 44%, 43%, and 41%, respectively. Furthermore, thanks to the contribution-wise byzantine-robust aggregation rule, Class-Balanced FL is able to assign significant weight to clients with high potential contributions in aggregation, ensuring a stable learning performance. In contrast, the global accuracy of FL Trust and Sageflow exhibits violent fluctuations, with an amplitude of up to 20% between the 35th and 60th learning iterations. For the attacked class, Class-Balanced FL achieves around 4% testing accuracy on Class 9, while all the existing FL methods achieve 0%. Figure 4.18 illustrates the global accuracy of Class-Balanced FL and different existing FL methods in cifa-10 training defending Class Imbalance attack; the inside subplot enlarges and demonstrates the pattern of learning performance between learning iteration 30th to 60th.

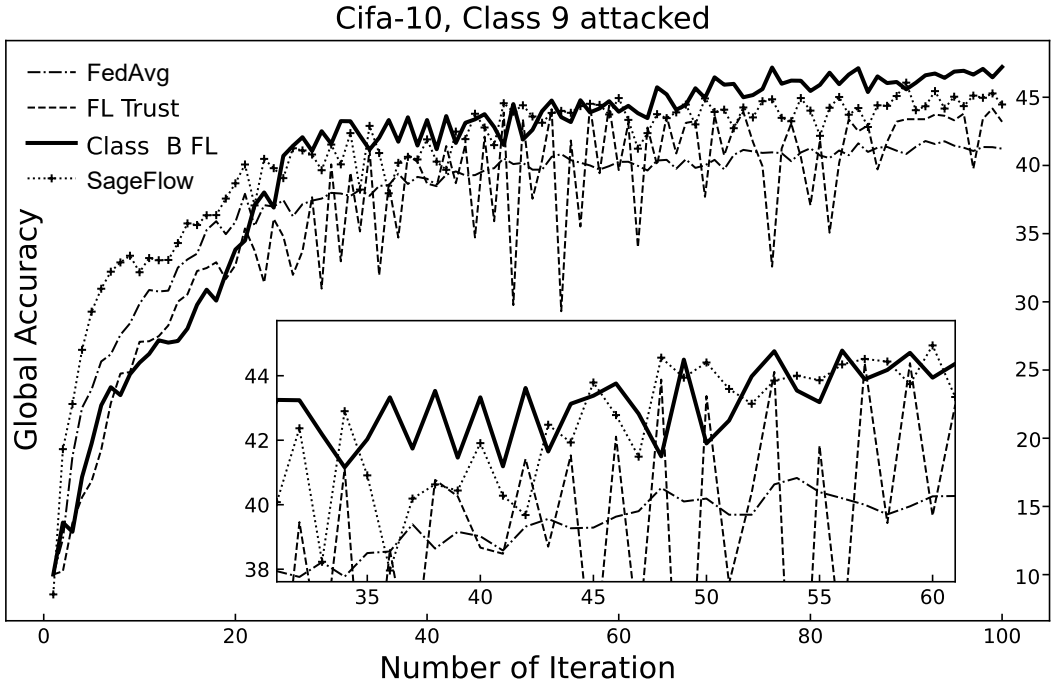

FIGURE 4.18: The illustration of the global accuracy of Class-Balanced FL and different FL methods in cifa-10 training defending Class Imbalance attack; the inside subplot enlarges and demonstrates the pattern of learning performance between learning iteration 30th to 60th.

4.2.5.4 Discussion

Now, we discuss why Class-Balanced FL can defend against the Class Imbalance attack. In the discussion of section III, we demonstrate the existing FL weight distribution algorithm is inflexible under the non-IID scenarios; the crafted gradient can constantly gain a heavyweight in the learning process even if the information carried has already been learned by the global model. Our Class-Balanced FL leverages the Contribution Score to enhance the flexibility of weight distribution. The client owning more class information can gain a heavyweight at the first several aggregation rounds. However, its weight will decrease due to the decreasing Contribution Score while its information is gradually learned by the global model, which enables the client owing fewer class data to participate in the learning task and enables the minor classes learned by the global model.

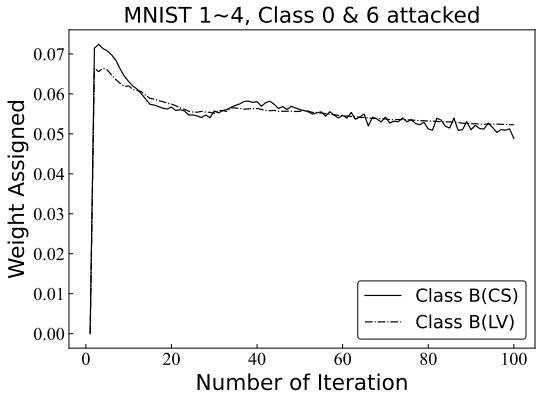

FIGURE 4.19: The weight gained by the attackers under CLass-Balanced FL.

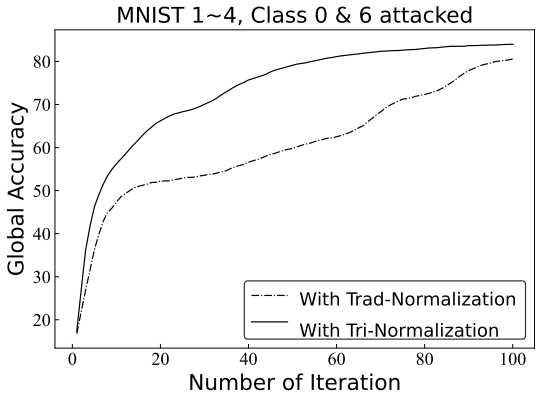

FIGURE 4.20: The comparison of model global accuracy with Tri-/Trad-Normalization.

Figure 4.19 illustrates the weight gained by each attacker under MNIST(1∼4) through Class-Balanced FL (CS) and (LV) when two labels are attacked. As they carry more class information, the attackers can gain a high Trust Score, Contribution Score, and a heavyweight at the start of the learning task (i.e., around 7%); but with the learning process, the weight gained decreases to 30% at 35 round due to the decreasing Contribution Score and finally around 5.5% after 60 round, which ensures other clients' information can be effectively aggregated in the following learning iterations.

We also compare the global accuracy of Class-Balanced FL (CS) using traditional normalization aggregation [16] and our proposed trimmed-normalization aggregation in the same scenario; experimental results reveal that Class-Balanced FL (CS) with trimmed-normalization aggregation achieves better learning performance. For instance, it achieves 75% and 83% accuracy at the 50th and 100th rounds, respectively, while achieving 56% and 80% accuracy with the traditional normalization aggregation method. The effectiveness of the trimmed-normalization aggregation method is depicted in Figure 4.20.

4.2.6 Conclusion

In this section, we proposed a new attack called the "Class Imbalance Attack" and demonstrated that the byzantine robust federated methods (which specifically keep a root dataset on the server side) are vulnerable to our attack. By manipulating the data distribution and uploading the corresponding gradient, the malicious client can effectively decrease the accuracy of the targeted class and subsequently decrease the global accuracy.

To defend against the Class Imbalance Attack and achieve byzantine robustness, we further proposed a new byzantine-robust FL method named "Class-Balanced" FL. The key difference between the Class-Balanced FL and the existing FL methods is that our Class-Balanced FL considers the current state of the global model, dynamically distributing the gradients' importance and deciding the magnitude of gradients in aggregation, which potentially brings more contribution. Our evaluation shows that the Class-Balanced FL is robust to the state-of-the-art poisoning attacks and the Class Imbalance Attacks.

CHAPTER 5

Balancing Implicit Class Learning Performance

In this chapter, we investigate the fairness issue of FL under non-IID scenarios, particularly when dealing with implicit classes. The dataset's inherent heterogeneity leads to one class of data containing various representations (i.e., explicit and implicit classes.) Such differences in the volume of training data for explicit and implicit classes can lead to imbalanced learning performance. To address this challenge, we propose the Single Class Training Scheme (SCTS) and the corresponding Implicit Class Balancing FL (ICB FL) algorithm, which identifies the implicit classes and fairly distributes weights during the aggregation process. We conduct evaluations using several datasets to demonstrate the performance of our proposed ICB FL. Our ICB FL framework was published at the 2023 Australasian Symposium on Parallel and Distributed Computing under the name ICB FL: Implicit Class Balancing Towards Fairness in Federated Learning.

5.1 Introduction

Federated Learning (FL) [160, 173] is a new machine learning paradigm that enables multiple clients to collaboratively train a model without sharing their raw data. In FL, the clients may have vastly different data volumes, various communication qualities, and data representations in light of the device, behavior, and preferences heterogeneity, which draws the non-independently identically distribution (i.e., non-IID) nature of the FL [65, 147, 180, 88]. Recent research [171] has indicated that the standard FL [97] can effectively aggregate the local model updates under the IID assumption while raising fairness issues in real-world non-IID scenarios. For instance, unfairness could exist in the "client selection" [145] and

"contribution evaluation" [100] stage if the client selection rule and contribution evaluation rule have been designed solely on the server's interest. As a result, some method-favored clients will have a higher probability to participate and contribute towards the learning task, while the "weaker" client might receive an unfair evaluation outcome and will be excluded.

Even worse, the fairness issue can occur during the global model optimization step when the class imbalance phenomenon occurs. Specifically, the classes (called majority classes) carried by more clients and account for a large ratio of overall training data can quickly achieve a high testing accuracy. In contrast, the classes owned by fewer clients (called minority classes) may receive a little participation weight and suffer from a slow learning speed. Research [170, 101] address this issue by optimizing the global/local objective and guaranteeing each class/client can gain a similar testing accuracy. However, these works [170, 101] only provide fairness across the explicit classes, but unfairness still exists among the implicit classes. An example is the lung cancer identification model, research [170, 101] enable "positive" and "negative" data items to achieve similar accuracy as they are explicit classes while failing to maintain the fairness between different genders as they are implicit classes and accounted for different volumes.

To mitigate the research gap, we propose the Implicit Class Balancing Federated Learning (ICB FL) framework and further introduce the Single Class Training Scheme (SCTS) to support the training process. We focus on the fairness of the global model optimization phase and manage to guarantee fairness across both explicit and implicit classes. Instead of using all data to train the local model, SCTS asks the client to train the local model through the data with one particular label. Because the gradient uploaded by the client only carries the information of one class (i.e., label) data, the server can subsequently identify the implicit class by performing unsupervised learning. At the end of each iteration, the server generates the weight for clients by balancing the weight across different explicit/implicit classes and weighted averages of the gradients. We evaluate our ICB FL on three datasets: Lung Cancer Survey, MNIST, and FMNIST. Our experimental results show the ICB FL guarantees the explicit classes level fairness and further receives a better implicit classes level fairness than [97, 101].

5.2 Related Work

As fairness issues could arise throughout the whole FL training process [87, 169], current research [57, 129, 100, 66, 170, 101] enhances the FL fairness from different perspectives. Work [57] considers enhancing fairness in the client selection phase by introducing a long-term fairness constraint. This constraint leverages a constant fairness parameter that prevents each client's average participation from being higher than (or similar to) the expected guaranteed rate. Similarly, work [129] also enhances the client selection fairness. [129] takes into account every client's real-time contribution and introduces a reputation-based client selection strategy. The server in [129] records the transmission history as the reputation and proposes a constraint to enable the client with both high and low reputation can be selected.

Research [100] addresses the fairness issue from the contribution evaluation perspective. Specifically, the server asks the participants to report the publicly verifiable factors of their local training data. The verifiable factors uploaded include data quality, amount, collection cost, etc. Based on the information reported, [100] subsequently assigns ratings to the participants. Similarly, the author [66] builds the incentive scheme based on the self-report. [66] leverages the Contract Theory [11] to design the contract items and broadcast them to all clients. The clients subsequently select the item most desired to participate in the FL learning task.

Besides the client selection and contribution evaluation phase, the severe unfairness issue also exists in the model optimization phase. This issue could result in the global model over-representing specific class data and exhibiting inequitable performance across participants. Research [170] tackles this issue by optimizing the global and local objectives to satisfy the expected fairness constraints and reduce the accuracy variance across clients. Author [101] provides a new FL method named AFL to prevent the global model from overfitting any particular client and achieve the good-intent notion of fairness. This work argues the distribution of clients may mismatch with the real target distribution due to the non-IID feature of FL and introduces the agnostic objective, which indicates the target distribution is unknown. Hence, in AFL [101], the server optimizes the global model for any distribution

formed by a mixture of clients. This work further proposes a new stochastic optimization solution to support the AFL and extend AFL to large-scale problems and several extensions.

5.3 Motivation

5.3.1 Problem Setup

Our problem is formulated on the synchronous federated learning classifier task; the classifier could be binary or multi-class. In each iteration of task training, the server first broadcasts the current global model M to n participants[1]; then, the participants locally train the model m and send the gradient[2] g back to the server. Finally, the server aggregates the gradients received through the aggregation method $f(\cdot)$ and optimizes the current global model M.

We use $x \in \mathcal{X}$ and y$\in \mathcal{Y}$ to denote the feature space and label space, respectively, where C is the total number of classes. We assume n participants jointly train the model M (i.e., classifier) through locally training a model $m_i, i = 1, 2, ..., n$ on their private data $p(x|y)$. The optimal model m_i^* is the solution for the following optimization problem of the participant i:

$$m_i^* = argmin_{m_i} F(m_i) \tag{5.1}$$

Here, $F(m_i) = \mathbb{E}_{(x,y)\sim p_i}[l((x, w_i), y)]$ represents the expectation of the empirical loss $l((x, g_i), y)$ and $p_i(x, y)$ is an private joint distribution of feature x_i and label y_i under the client i. The distribution $p_i(x, y)$ can be further composed as following,

$$p_i(x, y) = p(x|y)p_i(y) \tag{5.2}$$

where $p(x|y)$ denotes the conditional distribution of feature x given label y and $p_i(y)$ is the marginal distribution of label y on client i.

[1] In this work, we combined use "participants" and "clients" with the same meaning.

[2] In this work, we combined use "model update" and "gradient" with the same meaning.

However, because of the non-IID nature, both the conditional distribution $p(x|y)$ and the marginal distribution $p_i(y)$ could be significantly different among the participants in the FL context. Specifically, because of the heterogeneity of the client, clients can have different features (i.e., representations) under the same class (i.e., the difference of $p(x|y)$) or own a different number of classes of data (i.e., the difference of $p_i(y)$).

The difference between the local model w_i and the optimal local model m_i^* is calculated as the gradient g_i and then sent back to the server. Once receive the preset amount of gradient g_i from the client, the server will aggregate the local model updates via an aggregation method $f(\cdot)$ and subsequently use it to update the global model:

$$\begin{aligned} g_i &= m_i^* - m_i \\ M^{t+1} &= M^t + \eta \cdot f(g_i^t),\ i = 1, 2, ...n \end{aligned} \tag{5.3}$$

In this paper, we focus on the fairness issues in the global model optimization phase and introduce the fairness of two levels: the conditional distribution level fairness (short as F_{con}) and the marginal distribution level fairness (short as F_{mar}). Specifically, the F_{con} and F_{mar} reflect whether the different features of a class and whether the different classes are equally learned by the model, respectively.

We use the term 'Explicit Class' to define the different training labels that usually act as the classifier. We use the term 'Implicit Class' to define the different representations under one training label.

5.3.2 Motivation Example

Now, we provide an example to show the weaknesses and limitations of the existing fair enhance FL method. The example is set as a binary classification task; a server and 20 participants jointly train a lung cancer prediction model under the FL architecture [97].

A) Training Dataset

The dataset we use here is named "survey lung cancer," which includes thousands of samples, and each sample is recorded from 16 different perspectives: Gender, Age, Smoking, Yellow fingers, Anxiety, Peer pressure, Chronic Disease, Fatigue, Allergy, Wheezing, Alcohol, Coughing, Shortness of Breath, Swallowing Difficulty, Chest pain, Lung Cancer. The dataset naturally carries more male records; we further customize and adjust the dataset to consist of 75% male records and 25% female records.

We randomly distribute 80% to 20 different clients as the training data and reserve 20% as testing data ($\mathcal{D}_T$). To reflect the real-world situation, each client is only assigned one gender data. To evaluate the fairness of the model, we further divide the testing data by label (positive label data $\mathcal{D}_{TP}$ and negative label data $\mathcal{D}_{TN}$) and gender (male data $\mathcal{D}_{TM}$ and female data $\mathcal{D}_{TFe}$). We use accuracy as metrics; the accuracy difference between $\mathcal{D}_{TP}$ and $\mathcal{D}_{TN}$ reflects the model's marginal distribution level fairness (i.e., F_{mar}), and the difference between $\mathcal{D}_{TM}$ and $\mathcal{D}_{TFe}$ reflects the model's conditional distribution level fairness (i.e., F_{con}).

B) Training Model and Learning Setting

We use a deep neural network as the model, which includes 5 dense layers. The detail of the model is shown in table 5.1. We set the learning rate as 0.001, the SGD as the optimizer, and trained the model 100 rounds. We further use a representative FL method algorithm (i.e., AFL [101]) to guarantee fairness.

TABLE 5.1: Illustration of the training model.

Layer ID	Layer	Activation Function	Regular
Layer 1	Dense 32x	Relu	L1 Regular ($\gamma = 0.003$)
Layer 2	Dense 64x	Relu	L1 Regular ($\gamma = 0.003$)
Layer 3	Dense 128x	Relu	L1 Regular ($\gamma = 0.003$)
Layer 4	Dense 16x	Relu	L1 Regular ($\gamma = 0.003$)
Layer 5	Dense 1x	Sigmoid	N/A

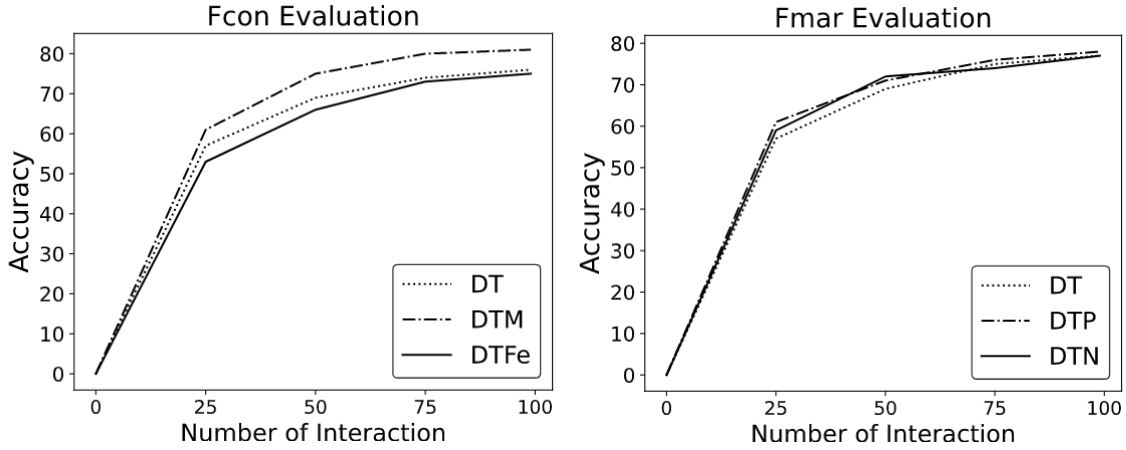

FIGURE 5.1: Illustration the F_{mar} and F_{con} of the lung cancer learning task.

C) Experiment Results

Figure 5.1 illustrates the F_{mar} and F_{con} of the FL model, respectively; it shows that the existing FL fairness method can achieve F_{mar} but fail to guarantee the F_{con}. Specially, all of $\mathcal{D}_T$, $\mathcal{D}_{TP}$ and $\mathcal{D}_{TN}$ achieve a very similar accuracy around 60%, 70%, 73% and 77% at round 25, 50, 75 and 100 respectively. However, under F_{con} evaluation, the $\mathcal{D}_{TM}$ achieves much higher testing accuracy than $\mathcal{D}_T$ and $\mathcal{D}_{TFe}$: $\mathcal{D}_{TM}$ achieves 73% and 83% at round 50 and 100, while $\mathcal{D}_T$ achieves 69% and 77%, $\mathcal{D}_{TFe}$ achieves 66% and 75% respectively. In other words, the positive and negative clients have a similar accuracy, while the male clients have a significantly higher accuracy than the female clients.

D) Discussion

Based on the experiment results, we note the existing fair FL [101] method can maintain fairness across different classes (i.e., labels) if they are explicit (e.g., positive or negative) but shows ineffectiveness when they are implicit (e.g., female and male) and account for different volumes.

Specifically, the explicit class indicates the label of each data item, which is usually used in the classification task and works as the classifier in the neural network. Because of the non-IID feature of the FL [65, 147, 180], each client may have a different number of data classes, which results in different marginal distributions. As these labels are explicit, the server can enhance their fairness by crafting the target labels' distribution of the learning

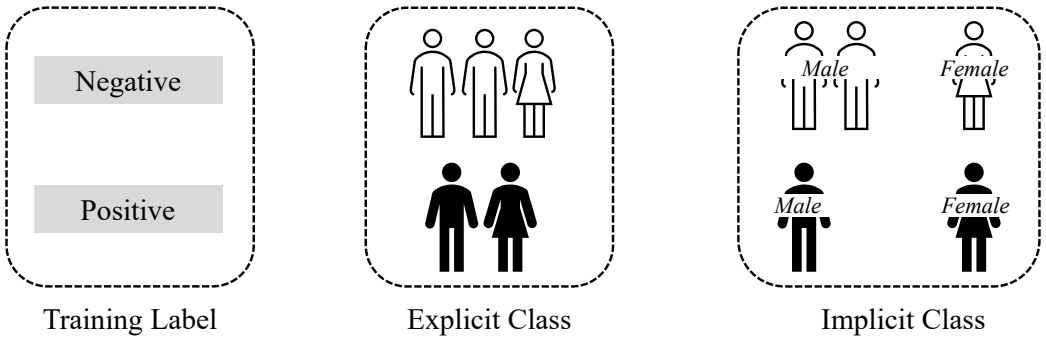

FIGURE 5.2: Illustration of the explicit/implicit class of lung cancer dataset.

object [101]. In contrast, the implicit class usually exists under the explicit class (i.e., label); a label may have different representations that refer to different implicit classes [65]. For instance, in the "lung cancer" example proposed, the positive label and negative labels are explicit classes, and the gender classes under each label are implicit; figure 5.2 illustrates the explicit/implicit classes of the "lung cancer" example. However, because these implicit groups are unseen, the object optimization-based fair FL method [101] can not guarantee fairness across the implicit groups.

5.4 Implicit Class Balancing Federated Learning

5.4.1 Overview of ICB FL

To mitigate the research gap and enhance the fairness of the existing federated learning [101], we propose the Implicit Class Balancing Federated Learning framework (short as ICB FL). The objectives of ICB FL are shown as follows:

I. Maintain the overall global accuracy: the ICB FL should maintain a similar global testing accuracy with the existing fair FL method [101].
II. Guarantee fairness (i.e., F_{mar}) across different explicit classes: different labels should have a similar testing accuracy under our ICB FL.
III. Guarantee fairness (i.e., F_{con}) across different implicit classes: the implicit classes should be identified and achieve a similar testing accuracy under our ICB FL.

The ICB FL considers a non-malicious scenario and leverages a new training scheme named Single Class Training Scheme (short as SCTS). Specifically, the server first broadcasts the current global model to all clients. Instead of using all local data to train the global model, each client will be assigned one particular class and start locally training only through the label c data. The gradients of the local model are subsequently sent back to the server, and the server performs unsupervised learning on each class gradient to detect the implicit class. Then, the server generates the weight for each implicit class (if it exists) to balance the contribution and participant. Finally, the server weighted averages the gradients of each implicit group generate each sub-gradient [3]; all sub-gradients are consequently averaged to be the new global gradient and used to update the global model through Equation (5.3).

The key difference between SCTS and the existing training scheme is the client only uses one label assigned to train the model. Because of the non-IID feature of FL, different explicit classes of data could be the leading resource of clustering; using one label to train the model can eliminate the effect of different classes owned and enables unsupervised learning to detect and identify the implicit classes.

5.4.2 ICB FL Framework

Our ICB FL relies on the proposed training (SCTS) scheme and includes five steps in each training round; Figure 5.3 illustrates the ICB FL.

Step I: Global Model Broadcast and Target Training Class Distribution

At the start of the training round t, the server broadcasts the current global model M^t to each client with a target training class $c, c \in C$. The class c is randomly generated by the server for each client, respectively. At the end of Step I, each client owns the M^t and specific target learning class $c, c = 1, 2, ..., C$.

Step II: Locally Training through SCTS

Once the client receives the current global model M^t and target class c, the client starts locally training the model through the Single Class Training Scheme. Specifically, each client locally

[3]We use "sub-gradient" to denote the gradient aggregated by all clients with the same trained label

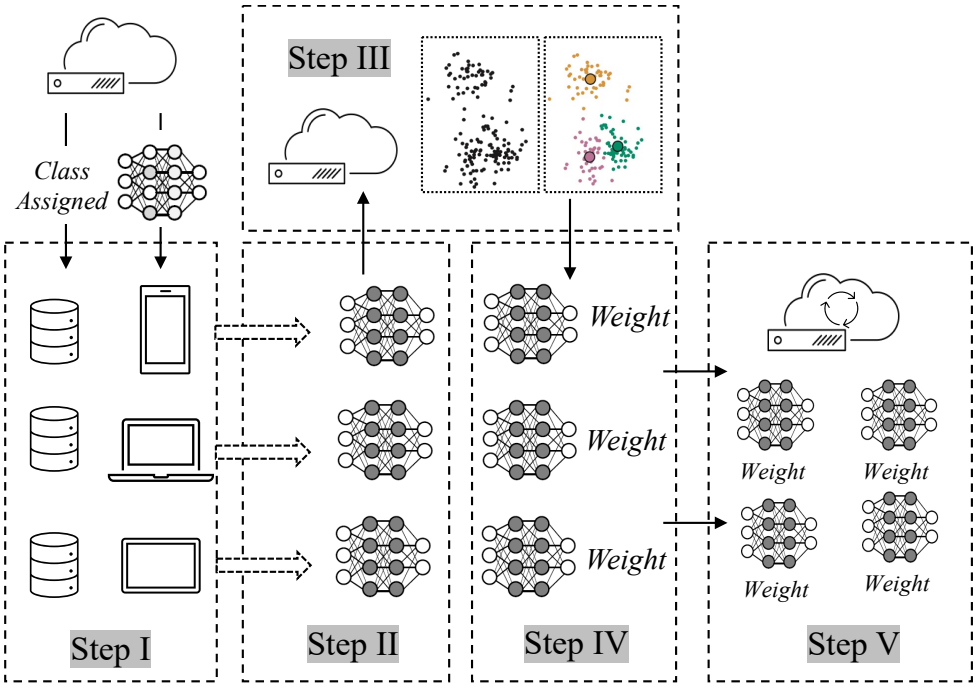

FIGURE 5.3: Illustration of the Implicit Class Balancing Federated Learning pipeline.

trains the M^t only through the class c training data. Once the preset local training round is achieved, the model updates ${}^{c}g_i^t, i = 1, 2, ...n$ are subsequently sent back to the server; here, c denotes the target training class for generating the gradient.

Step III: Implicit Groups Detection

The server performs the unsupervised learning for each class gradient received to detect the implicit groups. In this work, we use the "K-means method [49]" as the unsupervised learning method and the "elbow method [132]" to identify the number of clustering. At the end of Step III, each gradient will be further assigned an implicit group ID (i.e., ${}^{c}_{s}g_i^t$, where s indicates the implicit group ID under the class c).

Step IV: Weight Generation

Once the implicit groups have been detected, the server generates the weight for each implicit group. To guarantee each implicit group can be fairly represented (learned) by the model, the implicit groups under the same (explicit) class are assigned the same weight. Then, each explicit class is assigned the same weight of $\frac{1}{C}$ to fairly participate in the learning. Formally, we have the following:

$$
\begin{aligned}
{}^{c}W &= \frac{1}{C},\ c = 1, 2, ..., C \\
{}^{c}_{s}w &= \frac{1}{S},\ s = 1, 2, ..., S
\end{aligned}
\tag{5.4}
$$

where the uppercase W_c, and lowercase w_c^s denote the weight assigned for the (explicit) class c and each implicit group under class c.

Step V: Global Model Update

When updating the global model, the server firstly aggregates the gradient for each class through the following:

$$
{}^{c}g^{t} = \sum_{s=1}^{S} {}^{c}_{s}\overline{g}^{t} \cdot {}^{c}_{s}w \tag{5.5}
$$

where ${}^{c}_{s}\overline{g}^{t}$ is the mean of gradient which belong implicit group s and under class c, ${}^{c}g^{t}$ is the gradient generated for class c.

Finally, the server aggregates the gradient of each class to generate the global gradient g^t and update the global model through g^t:

$$
M^{t+1} = M^{t} + \eta \cdot g^{t}, \text{where } \ g^{t} = \sum_{c=1}^{C} {}^{c}g^{t} \cdot {}^{c}W \tag{5.6}
$$

We note that some classes may not have implicit classes when performing Step III, and the gradient can not be divided by unsupervised learning. In this case, the class c does not need to generate the implicit class weights (i.e., ${}^{c}_{s}w$) and can directly perform the mean method for the associated gradient to generate ${}^{c}g^{t}$ and participates Step V.

5.4.3 Fairness Enhancement Discussion

Here, we provide the discussion of our proposed ICB FL from fairness guarantee and privacy guarantee two perspectives.

ICB FL guarantees the fairness of FL by crafting the weight of each explicit and implicit class. By balancing the weight, ICB FL can eliminate the amount of difference impact of the different explicit/implicit class data and enables them to be similarly learned and represented by the global model. Furthermore, by leveraging the SCTS and unsupervised learning, ICB FL does not require the client to provide further information instead of the gradient for implicit class identification. Furthermore, based on the clustering result, the server can not reverse infer the resource for clustering (e.g., "Gender" in the motivation example), which maintains the data minimization principle [65] of FL.

5.5 Evaluation

We evaluate the fairness of our ICB FL from F_{mar} and F_{con} two perspectives. We use the range (i.e., R [4]) of the testing accuracy as the metrics to reflect the fairness; specifically, different implicit/explicit classes should achieve a similar testing accuracy (a small range R). We use the Fedavg [97] as the baseline and introduce the AFL method [101] to compare with.

5.5.1 Experimental setup

We follow the setting of Section 5.3 and assume 20 clients participate in the learning task. Besides the "lung cancer survey" dataset, we further leverage MNIST [31] and FMNIST [152] for evaluation. As these datasets only [31, 152] contain the one layer class (training label), we further modified these datasets to guarantee both explicit and implicit are contained. Besides, we filter and clean the dataset to mitigate the learning performance gap between classes caused by training data quality. We name the modified MNIST as $MNIST_{modi}$, and Fashion MNIST as $FMNIST_{modi}$.

A) Training Data Processing

$MNIST_{modi}$ is generated based on the MNIST, which is an extensive database of handwritten digits that includes 60,000 training images and 10,000 testing images; each image has consisted of 28×28 pixels and a label of $0 \sim 9$ to stand for the class belonging to. We

[4] R: range, the difference between the highest and the lowest values in a set.

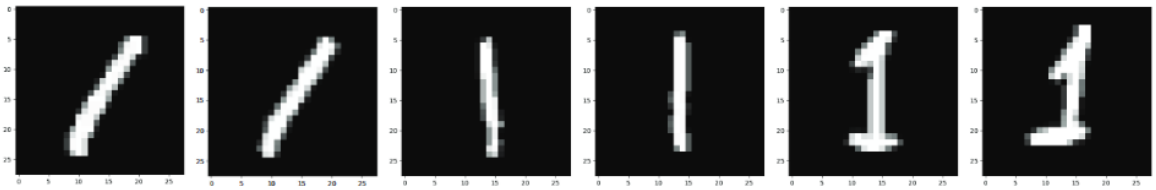

FIGURE 5.4: Illustration of the different representations of label 1 under MNIST: italic font (left two), normal font (middle two), floral font (right two).

note that label 1 includes three different representations (italic font, normal font, and floral font), making up 75%, 20%, and 5% overall label 1 data. Figure 5.4 illustrates the different representations of label 1 under MNIST. We distribute the 10 explicit classes (with a similar amount) and 3 implicit classes (with different amounts) training data to 20 clients following the parameter p, where p is the percentage of the class of data. To simulate the real-world scenario, one client can be assigned different explicit classes but only one implicit class training data. We use the original testing data to evaluate the overall accuracy of the model, split the testing data into 10 groups by the label (i.e., $\mathcal{D}_{T0} \sim \mathcal{D}_{T9}$) to evaluate the accuracy on each explicit class and F_{mar}, and split the label 1 testing data into 3 groups by the font (i.e., $\mathcal{D}_{TIta}$, $\mathcal{D}_{TNor}$, $\mathcal{D}_{TFlo}$) to evaluate the accuracy on each implicit class and F_{con}.

$FMNIST_{modi}$ is generated based on the Fashion MNIST and includes 60,000 gray-scale images of 10 fashion categories and a test set of 10,000 images. To simulate the explicit/implicit class scenario, we use the original 10 labels as the implicit class and further introduce 4 explicit class (i.e., tops, bottoms, shoes, and accessories) as the training labels above them. Table 5.2 illustrates the explicit/implicit class of $FMNIST_{modi}$.

We reserve the original amount of T-shirt, Coat, Pullover, Trouser, Sandal, and Sneaker training data and reduce the other training data by half to simulate the different amounts of representations in the real world. Because including different implicit classes, Tops (40%) and Shoes (30%) carry more data than Bottoms (18%) and Accessories (12%). We follow the $MNIST_{modi}$ data distribution and distribute the $FMNIST_{modi}$ training data by p. Furthermore, we use the original testing data (with new labels) to evaluate the global accuracy of the model and split the testing data by explicit (i.e., $\mathcal{D}_{TTop} \sim \mathcal{D}_{TAcc}$) and implicit classes (i.e., $\mathcal{D}_{TT-shirt} \sim \mathcal{D}_{TBag}$) to evaluate the model's F_{mar} and F_{con} respectively. As the T-shirt,

Coat, Pullover, Trouser, Sandal, and Sneaker (called major class) carry a similar original amount of training data and teasing results, we use $\mathcal{D}_{TMaj}$ to generally denote the class of them for the sake of briefness. Similarly, we use $\mathcal{D}_{TMin}$ to denote the class (called minor class)in which the training data has been reduced by half.

TABLE 5.2: Illustration of Explicit and Implicit class of $FMNIST_{modi}$.

Explicit Class	Tops	Bottoms	Shoes	Accessories
Implicit Class	T-shirt, Coat, Pullover, Shirt	Trouser Dress	Sandal, Sneaker Ankle boot	Bag

B) Training Model

Expect the model we introduced in the previous section for training the lung cancer prediction model; we further introduce a general model for $MNIST_{modi}$ and $FMNIST_{modi}$ training tasks; the model consists of a dense layer (28×28) and a softmax layer ($\times 10$). We set the learning rate as 0.01, the batch size as 128, and the epoch as 50 to train the global model 100 and 300 iterations for $MNIST_{modi}$ and $FMNIST_{modi}$, respectively.

5.5.2 Experimental Results

Recall our objectives set in Section 5.4.1 are maintaining the overall global accuracy and enhancing the F_{mar} and F_{con} of the existing FL methods; the experimental results show our ICB FL achieves all three objectives.

First, ICB FL achieves a similar global accuracy compared to the current FL method. Specifically, ICB FL receives around 77%, 84% and 85% in Lung Cancer Prediction, $MNIST_{modi}$ and $FMNIST_{modi}$ evaluation respectively, while FedAvg [97] achieves 77%, 86% and 86%, and AFL [101] achieves 77%, 87% and 88%. We note that the global accuracy of ICB FL is slightly lower than Fedavg and AFL; this is because ICB FL clients use only positive (only one label) data to train their model, which may introduce trivial solutions [168] and decrease the global model accuracy. Figure 5.5 illustrates the global accuracy of Fedavg, AFL, and ICB FL methods under different datasets.

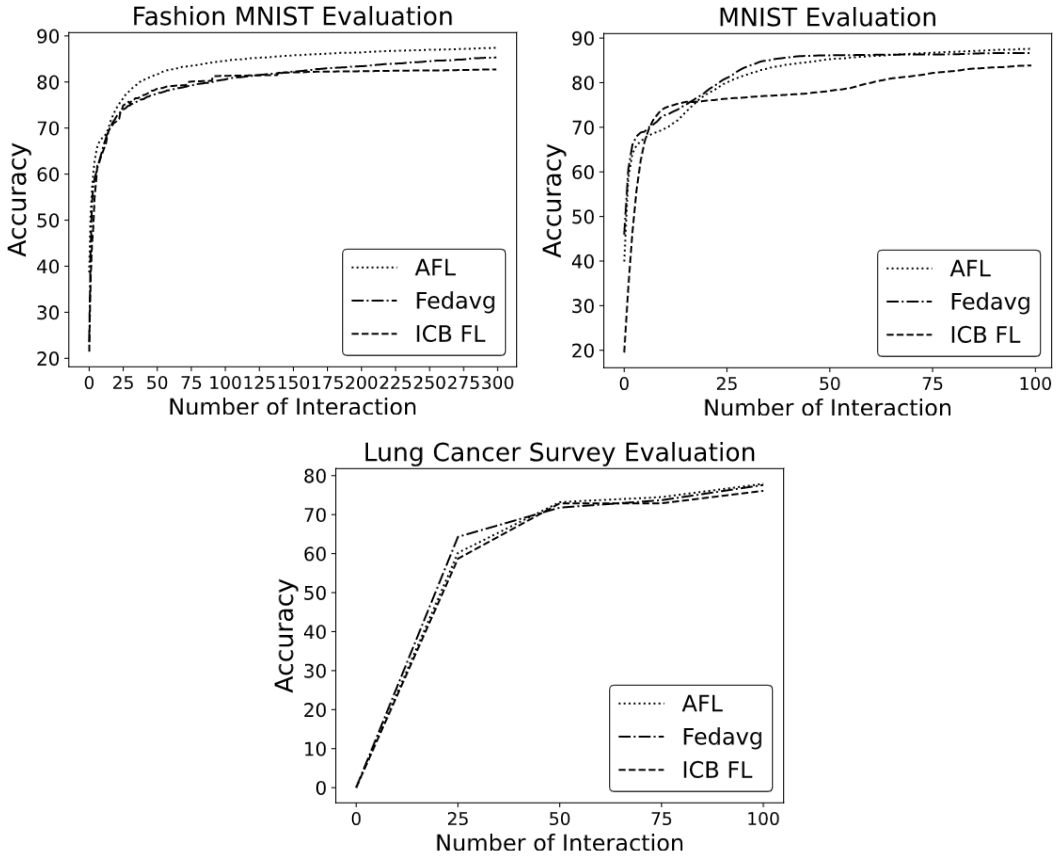

FIGURE 5.5: Illustration of the global accuracy of AFL/Fedavg/ICB FL under different datasets.

Second, ICB FL guarantees F_{mar} (i.e., marginal distribution level fairness) for the FL learning task and enables each explicit class to receive a similar accuracy. For instance, different explicit classes achieve testing accuracy of $75.1\% \sim 78.2\%$ under Lung Cancer Prediction (R=3.1%) $80.1\% \sim 84.7\%$ under $MNIST_{modi}$ (R=4.6%) and $83.2\% \sim 85.6\%$ (R=2.4%) under $FMNIST_{modi}$, respectively. We note that AFL receives a similar accuracy range with R=2.4%, 3.6%, and 3.2% while Fedavg fails to guarantee the F_{mar} and draws a big accuracy range. Table 5.3 illustrates the testing accuracy across different explicit classes, showing that our ICB FL and AFL methods effectively mitigate the negative influence carried by different explicit class participants and enable the global model to achieve balance learning performance across different explicit classes.

Third, our ICB FL framework effectively identifies the implicit classes, enhances the F_{con} (i.e., conditional distribution level fairness) for the existing FL methods, and reduces the accuracy

TABLE 5.3: Illustration of the testing accuracy of different explicit classes.

Dataset	Lung Cancer	$MNIST_{modi}$	$FMNIST_{modi}$
Explicit Class	$\mathcal{D}_{TP}$, $\mathcal{D}_{TN}$	$\mathcal{D}_{T0} \sim \mathcal{D}_{T9}$	$\mathcal{D}_{TTop} \sim \mathcal{D}_{TAcc}$
FedAvg	$74.4\% \sim 77.9\%$	$82.3\% \sim 86.9\%$	$78.5\% \sim 87.4\%$
AFL	$75.7\% \sim 78.1\%$	$83.8\% \sim 87.4\%$	$85.2\% \sim 88.4\%$
ICB FL	$75.1\% \sim 78.2\%$	$80.1\% \sim 84.7\%$	$83.2\% \sim 85.6\%$

gap (i.e., range) between different implicit classes. Table 5.4 illustrates the testing accuracy across different implicit classes; the results under $\mathcal{D}_{TMaj}$ and $\mathcal{D}_{TMin}$ are the average testing accuracy of all major and minor classes. Specially, ICB FL achieves 5.1%, 10.1%, and 5.9% accuracy range across different implicit classes under Lung Cancer Prediction, $MNIST_{modi}$ and $FMNIST_{modi}$; while AFL and Fedavg fail to guarantee the F_{con} which achieve R as 7.7% and 6.6%, 18.8% and 17.1%, 11.6% and 10.5% in three datasets, respectively.

TABLE 5.4: Illustration of the testing accuracy of different implicit classes.

Dataset	Lung Cancer		$MNIST_{modi}$			$FMNIST_{modi}$	
Implicit Class	$\mathcal{D}_{TM}$	$\mathcal{D}_{TFe}$	$\mathcal{D}_{TIta}$	$\mathcal{D}_{TNor}$	$\mathcal{D}_{TFlo}$	$\mathcal{D}_{TMaj}$	$\mathcal{D}_{TMin}$
FedAvg	82.3%	75.7%	86.4%	80.1%	69.3%	87.8%	77.3%
AFL	83.1%	75.4%	87.5%	75.6%	68.7%	88.4%	76.8%
ICB FL	81.8%	76.7%	85.8%	77.2%	75.7%	85.7%	79.8%

5.6 Conclusion

In this chapter, we propose a new FL framework named Implicit Class Balancing Federated Learning with a new training scheme named Single Class Training Scheme to enhance the fairness of federated learning. Our experimental results show the ICB FL can effectively identify the implicit classes and guarantee both implicit level and explicit level fairness through wise weight distribution. We note that our ICB FL slightly decreases the global accuracy

because the current training scheme introduces trivial solutions when client separately trains their models with only one (positive) label data. The future work includes investigating how to effectively train the model and aggregate the gradient generated by the only positive label training data and extending our ICB FL to more complicated classification scenarios.

CHAPTER 6

Conclusion and Future Work

In this thesis, we study several topics on the robustness and fairness of federated learning (FL) in non-IID scenarios, including attack surface reduction, poisoning attack defense, and implicit class-level fair enhancement. In particular, we identify the main non-IID resources of FL as Geo-feature, Time-feature, and User-feature. Based on this insight, we propose the Mini-FL framework, which assigns the local model updates to different groups and achieves attack surface reduction. To defend against Label Flipping Attacks, we propose HSCS FL. In each learning iteration, HSCS FL assesses the accuracy of each individual class in both the global and local models. These accuracy measurements are then translated into an honest score; only the clients achieving the top scores are included in the following aggregation. To defend against the Class Imbalance Attack, we propose Class-Balanced FL, which dynamically decides the aggregation weight of each client based on their potential contribution to the current global model. To enhance FL fairness, we propose ICB FL to identify implicit classes and make wise weight distributions to guarantee they achieve a similar learning performance.

For each proposed scheme and framework, we provide mathematical proven and carry out experiments and simulations to evaluate their performance, where they all demonstrate a significant improvement compared to the benchmarks and bring significant benefits to both FL robustness and fairness.

Although we provide some progress in this thesis, constructing a robust and fair FL environment remains an ongoing endeavor. Here, we discuss some research directions that could be explored in the future:

First, due to the non-IID data distribution, the FL participants could possess uncommon or low-quality data. Even though these clients have benign intentions, they are often deemed suspicious and excluded from the aggregation due to subpar local model performance. How to effectively aggregate benign clients with disadvantaged training resources and distinguish them from malicious clients could be an important research direction. We consider that a meticulously designed optimizer-level watermark could serve as a viable solution, wherein any direct manipulation of the gradients would compromise the integrity of the watermark.

Second, FL depends on regular communication between the server and clients to aggregate and update the global model. This communication process opens new avenues for adversaries to exploit. Given that the current cybersecurity research in FL is still nascent, investing in these communication-level FL vulnerabilities and devising defense frameworks are important to enhance FL robustness.

Finally, while we pinpoint geography, time, and client heterogeneity as the primary non-IID resources in FL, the origins of non-IID can be more intricate in specific settings and, in some instances, even be a combination of factors. Delving deeper into the non-IID sources in FL presents an intriguing avenue . In future research, we consider starting from identify various use cases of FL. Subsequently, we plan to employ unsupervised learning techniques to meticulously investigate each scenario individually.

Bibliography

[1] Kim Abildgren. “Archival big data and the spanish flu in copenhagen.” In: *Information Discovery and Delivery* 50.2 (2022), pp. 133–141.

[2] Sarita Agrawal, Manik Lal Das, and Javier Lopez. “Detection of node capture attack in wireless sensor networks.” In: *IEEE Systems Journal* 13.1 (2018), pp. 238–247.

[3] Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. “Differentially private learning with adaptive clipping.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2021, pp. 17455–17466.

[4] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. “How to backdoor federated learning.” In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR. 2020, pp. 2938–2948.

[5] Abolfazl Baghbani, Tanveer Choudhury, Susanga Costa, and Johannes Reiner. “Application of artificial intelligence in geotechnical engineering: A state-of-the-art review.” In: *Earth-Science Reviews* 228.1 (2022), pp. 1–26.

[6] Gilad Baruch, Moran Baruch, and Yoav Goldberg. “A little is enough: Circumventing defenses for distributed learning.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2019, pp. 1–11.

[7] Paolo Bellavista, Luca Foschini, and Alessio Mora. “Decentralised learning in federated deployment environments: A system-level survey.” In: *ACM Computing Surveys* 54.1 (2021), pp. 1–38.

[8] Battista Biggio, Blaine Nelson, and Pavel Laskov. “Poisoning attacks against support vector machines.” In: *International Coference on International Conference on Machine Learning (ICML)*. ACM. 2012, pp. 1467–1474.

[9] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. “Machine learning with adversaries: Byzantine tolerant gradient descent.” In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2017, pp. 1–11.

[10] Alberto Blanco-Justicia, Josep Domingo-Ferrer, Sergio Martınez, David Sánchez, Adrian Flanagan, and Kuan Eeik Tan. “Achieving security and privacy in federated learning systems: Survey, research challenges and future directions.” In: *Engineering Applications of Artificial Intelligence* 106.1 (2021), pp. 1–14.

[11] Patrick Bolton and Mathias Dewatripont. *Contract theory*. MIT press, 2004.

[12] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. “Adaptive Federated Dropout: Improving Communication Efficiency and Generalization for Federated Learning.” In: *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2020, pp. 1–6.

[13] Wieland Brendel, Jonas Rauber, and Matthias Bethge. “Decision-based adversarial attacks: Reliable attacks against black-box machine learning models.” In: *International Conference on Learning Representations (ICLR)*. IMLS. 2018, pp. 1–17.

[14] Christopher Briggs, Zhong Fan, and Peter Andras. “Federated learning with hierarchical clustering of local updates to improve training on non-IID data.” In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–9.

[15] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. “Understanding distributed poisoning attack in federated learning.” In: *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE. 2019, pp. 233–239.

[16] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. “FLTrust: Byzantine-robust federated learning via trust bootstrapping.” In: *ISOC Network and Distributed System Security Symposium (NDSS)*. ISOC. 2021, pp. 1–18.

[17] Xiaoyu Cao and Neil Zhenqiang Gong. “Mpaf: Model poisoning attacks to federated learning based on fake clients.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2022, pp. 3396–3404.

[18] Xiaoyu Cao, Zaixi Zhang, Jinyuan Jia, and Neil Zhenqiang Gong. “Flcert: Provably secure federated learning against poisoning attacks.” In: *IEEE Transactions on Information Forensics and Security* 17 (2022), pp. 3691–3705.

[19] Xinyang Cao and Lifeng Lai. “Distributed gradient descent algorithm robust to an arbitrary number of byzantine attackers.” In: *IEEE Transactions on Signal Processing* 67.22 (2019), pp. 5850–5864.

[20] Cheng Chen, Bhavya Kailkhura, Ryan Goldhahn, and Yi Zhou. “Certifiably-robust federated adversarial learning via randomized smoothing.” In: *IEEE International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE. 2021, pp. 173–179.

[21] Lingjiao Chen, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. “Draco: Byzantine-resilient distributed training via redundant gradients.” In: *International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 903–912.

[22] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. “Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models.” In: *ACM Workshop on Artificial Intelligence and Security (AISec)*. ACM. 2017, pp. 15–26.

[23] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li. “Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach.” In: *Computers & Security* 73 (2018), pp. 326–344.

[24] Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. “Asynchronous online federated learning for edge devices with non-iid data.” In: *IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, pp. 15–24.

[25] Yizong Cheng. “Mean shift, mode seeking, and clustering.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995), pp. 790–799.

[26] Ellen Collins and Graham Stone. “Understanding patterns of library use among undergraduate students from different disciplines.” In: *Evidence Based Library and Information Practice* 9.3 (2014), pp. 51–67.

[27] Gabriela F. Cretu, Angelos Stavrou, Michael E. Locasto, Salvatore J. Stolfo, and Angelos D. Keromytis. “Casting out Demons: Sanitizing Training Data for Anomaly Sensors.” In: *IEEE Symposium on Security and Privacy (SP)*. IEEE. 2008, pp. 81–95.

[28] Sen Cui, Weishen Pan, Jian Liang, Changshui Zhang, and Fei Wang. “Addressing algorithmic disparity and performance inconsistency in federated learning.” In: *Advances*

in Neural Information Processing Systems (NeurIPS). MIT Press. 2021, pp. 26091–26102.

[29] Georgios Damaskinos, El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, and Sébastien Rouault. "Aggregathor: Byzantine machine learning via robust gradient aggregation." In: *Machine Learning and Systems (MLSys)*. ACM. 2019, pp. 81–106.

[30] Deepesh Data, Linqi Song, and Suhas N. Diggavi. "Data encoding for byzantine-resilient distributed optimization." In: *IEEE Transactions on Information Theory* 67.2 (2021), pp. 1117–1140.

[31] Li Deng. "The mnist database of handwritten digit images for machine learning research." In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[32] Georgios Drainakis, Konstantinos V Katsaros, Panagiotis Pantazopoulos, Vasilis Sourlas, and Angelos Amditis. "Federated vs. centralized machine learning under privacy-elastic users: A comparative analysis." In: *IEEE International Symposium on Network Computing and Applications (NCA)*. IEEE. 2020, pp. 1–8.

[33] Wei Du, Depeng Xu, Xintao Wu, and Hanghang Tong. "Fairness-aware agnostic federated learning." In: *SIAM International Conference on Data Mining (SDM)*. SIAM. 2021, pp. 181–189.

[34] Moming Duan, Duo Liu, Xianzhang Chen, Renping Liu, Yujuan Tan, and Liang Liang. "Self-balancing federated learning with global imbalanced data in mobile systems." In: *IEEE Transactions on Parallel and Distributed Systems* 32.1 (2020), pp. 59–71.

[35] Gamaleldin Elsayed, Shreya Shankar, Brian Cheung, Nicolas Papernot, Alexey Kurakin, Ian Goodfellow, and Jascha Sohl-Dickstein. "Adversarial examples that fool both computer vision and time-limited humans." In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2018, pp. 1–11.

[36] Yahya H Ezzeldin, Shen Yan, Chaoyang He, Emilio Ferrara, and A Salman Avestimehr. "Fairfed: Enabling group fairness in federated learning." In: *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI. 2023, pp. 7494–7502.

[37] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. "Local model poisoning attacks to byzantine-robust federated learning." In: *USENIX Security Symposium (USENIX Security)*. USENIX Association. 2020, pp. 1605–1622.

[38] Lei Feng, Yiqi Zhao, Shaoyong Guo, Xuesong Qiu, Wenjing Li, and Peng Yu. "BAFL: A blockchain-based asynchronous federated learning framework." In: *IEEE Transactions on Computers* 71.5 (2022), pp. 1092–1103.

[39] Yann Fraboni, Richard Vidal, and Marco Lorenzi. "Free-rider attacks on model aggregation in federated learning." In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR. 2021, pp. 1846–1854.

[40] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. "An efficient framework for clustered federated learning." In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2020, pp. 19586–19597.

[41] Xueluan Gong, Yanjiao Chen, Qian Wang, and Weihan Kong. "Backdoor attacks and defenses in federated learning: State-of-the-art, taxonomy, and future directions." In: *IEEE Wireless Communications* 30.2 (2023), pp. 114–121.

[42] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." In: *International Conference on Learning Representations (ICLR)*. IMLS. 2015, pp. 1–11.

[43] Rémi Gosselin, Loıc Vieu, Faiza Loukil, and Alexandre Benoit. "Privacy and security in federated learning: A survey." In: *Applied Sciences* 12.19 (2022), pp. 1–15.

[44] Rachid Guerraoui, Arsany Guirguis, Jérémy Plassmann, Anton Ragot, and Sébastien Rouault. "Garfield: System support for byzantine machine learning (regular paper)." In: *International Conference on Dependable Systems and Networks (DSN)*. IEEE. 2021, pp. 39–51.

[45] Rachid Guerraoui, Sébastien Rouault, et al. "The hidden vulnerability of distributed learning in byzantium." In: *International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 3521–3530.

[46] Himanshu Gupta and Om Prakash Verma. "Monitoring and surveillance of urban road traffic using low altitude drone images: a deep learning approach." In: *Multimedia Tools and Applications* 81.1 (2022), pp. 1–21.

[47] Xu Han, Haoran Yu, and Haisong Gu. "Visual inspection with federated learning." In: *Image Analysis and Recognition: International Conference, (ICIAR)*. Springer. 2019, pp. 52–64.

[48] David Harrison Jr and Daniel L Rubinfeld. “Hedonic housing prices and the demand for clean air.” In: *Journal of Environmental Economics and Management* 5.1 (1978), pp. 81–102.

[49] John A Hartigan and Manchek A Wong. “Algorithm AS 136: A k-means clustering algorithm.” In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2016, pp. 770–778.

[51] Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. “Federated robustness propagation: Sharing adversarial robustness in heterogeneous federated learning.” In: *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI. 2023, pp. 7893–7901.

[52] Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. “The non-iid data quagmire of decentralized machine learning.” In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 4387–4398.

[53] Binxuan Hu, Yujia Gao, Liang Liu, and Huadong Ma. “Federated region-learning: An edge computing based framework for urban environment sensing.” In: *IEEE Global Communications Conference (GlobeCom)*. IEEE. 2018, pp. 1–7.

[54] Zeou Hu, Kiarash Shaloudegi, Guojun Zhang, and Yaoliang Yu. “Federated learning meets multi-objective optimization.” In: *IEEE Transactions on Network Science and Engineering* 9.4 (2022), pp. 2039–2051.

[55] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. “Data poisoning attacks to deep learning based recommender systems.” In: *Network and Distributed System Security Symposium (NDSS)*. 2021, pp. 1–17.

[56] Li Huang, Andrew L Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. “Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records.” In: *Journal of Biomedical Informatics* 99 (2019), pp. 1–15.

[57] Tiansheng Huang, Weiwei Lin, Wentai Wu, Ligang He, Keqin Li, and Albert Y Zomaya. “An efficiency-boosting client selection scheme for federated learning with

fairness guarantee." In: *IEEE Transactions on Parallel and Distributed Systems* 32.7 (2020), pp. 1552–1564.

[58] Yangsibo Huang, Samyak Gupta, Zhao Song, Kai Li, and Sanjeev Arora. "Evaluating gradient inversion attacks and defenses in federated learning." In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2021, pp. 7232–7241.

[59] Ahmed Imteaj, Urmish Thakker, Shiqiang Wang, Jian Li, and M Hadi Amini. "A survey on federated learning for resource-constrained IoT devices." In: *IEEE Internet of Things Journal* 9.1 (2021), pp. 1–24.

[60] Sohei Itahara, Takayuki Nishio, Yusuke Koda, Masahiro Morikura, and Koji Yamamoto. "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data." In: *IEEE Transactions on Mobile Computing* 22.1 (2021), pp. 191–205.

[61] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. "Manipulating machine learning: Poisoning attacks and countermeasures for regression learning." In: *IEEE Symposium on Security and Srivacy (SP)*. IEEE. 2018, pp. 19–35.

[62] Feiyu Jiang, Zifeng Zhao, and Xiaofeng Shao. "Time series analysis of COVID-19 infection curve: A change-point perspective." In: *Journal of Econometrics* 232.1 (2023), pp. 1–17.

[63] Wenbo Jiang, Hongwei Li, Sen Liu, Yanzhi Ren, and Miao He. "A flexible poisoning attack against machine learning." In: *IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–6.

[64] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. "Model pruning enables efficient federated learning on edge devices." In: *IEEE Transactions on Neural Networks and Learning Systems* 1.1 (2022), pp. 1–13.

[65] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. "Advances and open problems in federated learning." In: *Foundations and Trends® in Machine Learning* 14.1–2 (2021), pp. 1–210.

[66] Jiawen Kang, Zehui Xiong, Dusit Niyato, Han Yu, Ying-Chang Liang, and Dong In Kim. "Incentive design for efficient federated learning in mobile networks: A contract theory approach." In: *IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*. IEEE. 2019, pp. 1–5.

[67] Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. "Reliable federated learning for mobile networks." In: *IEEE Wireless Communications* 27.2 (2020), pp. 72–80.

[68] Tung Kieu, Bin Yang, Chenjuan Guo, Razvan-Gabriel Cirstea, Yan Zhao, Yale Song, and Christian S Jensen. "Anomaly detection in time series with robust variational quasi-recurrent autoencoders." In: *IEEE International Conference on Data Engineering (ICDE)*. IEEE. 2022, pp. 1342–1354.

[69] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S Jensen. "Outlier Detection for Time Series with Recurrent Autoencoder Ensembles." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann. 2019, pp. 2725–2732.

[70] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. "Blockchained on-device federated learning." In: *IEEE Communications Letters* 24.6 (2019), pp. 1279–1283.

[71] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. "Federated tensor factorization for computational phenotyping." In: *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM. 2017, pp. 887–895.

[72] Yeongwoo Kim, Ezeddin Al Hakim, Johan Haraldson, Henrik Eriksson, José Mairton B da Silva, and Carlo Fischione. "Dynamic clustering in federated learning." In: *IEEE International Conference on Communications (ICC)*. IEEE. 2021, pp. 1–6.

[73] Trupti M Kodinariya and Prashant R Makwana. "Review on determining number of cluster in k-means clustering." In: *International Journal* 1.6 (2013), pp. 90–95.

[74] Pang Wei Koh, Jacob Steinhardt, and Percy Liang. "Stronger data poisoning attacks break data sanitization defenses." In: *Machine Learning* 111.1 (2022), pp. 1–47.

[75] A Krizhevsky. "Learning multiple layers of features from tiny images." In: *Master's thesis, University of Tront* (2009).

[76] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. “Adversarial machine learning at scale.” In: *International Conference on Learning Representations (ICLR)*. IMLS. 2017, pp. 1–17.

[77] Gopal Singh Kushwah and Virender Ranga. “Detecting DDoS attacks in cloud computing using extreme learning machine and adaptive differential evolution.” In: *Wireless Personal Communications* 124.3 (2022), pp. 2613–2636.

[78] Cody Lewis, Vijay Varadharajan, and Nasimul Noman. “Attacks against federated learning defense systems and their mitigation.” In: *Journal of Machine Learning Research* 24.30 (2023), pp. 1–50.

[79] Chengxi Li, Gang Li, and Pramod K. Varshney. “Federated Learning With Soft Clustering.” In: *IEEE Internet of Things Journal* 9.10 (2022), pp. 7773–7782.

[80] Dongcheng Li, W. Eric Wong, Wei Wang, Yao Yao, and Matthew Chau. “Detection and Mitigation of Label-Flipping Attacks in Federated Learning Systems with KPCA and K-Means.” In: *International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2021, pp. 551–559.

[81] Jie Li, Tianqing Zhu, Wei Ren, and Kim-Kwang Raymond. “Improve individual fairness in federated learning via adversarial training.” In: *Computers & Security* (2023), pp. 1–10.

[82] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. “A review of applications in federated learning.” In: *Computers & Industrial Engineering* 149.1 (2020), pp. 1–15.

[83] Liping Li, Wei Xu, Tianyi Chen, Georgios B Giannakis, and Qing Ling. “RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets.” In: *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI. 2019, pp. 1544–1551.

[84] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. “Federated learning on non-iid data silos: An experimental study.” In: *International Conference on Data Engineering (ICDE)*. IEEE. 2022, pp. 965–978.

[85] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. “A survey on federated learning systems: vision, hype and reality

for data privacy and protection." In: *IEEE Transactions on Knowledge and Data Engineering* 35.1 (2021), pp. 3347–3366.

[86] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. "Federated optimization in heterogeneous networks." In: *Machine Learning and Systems (MLSys)*. ACM. 2020, pp. 429–450.

[87] Tian Li, Maziar Sanjabi, Ahmad Beirami, and Virginia Smith. "Fair Resource Allocation in Federated Learning." In: *International Conference on Learning Representations (ICLR)*. IMLS. 2019, pp. 1–27.

[88] Yanli Li, Abubakar Sadiq Sani, Dong Yuan, and Wei Bao. "Enhancing federated learning robustness through clustering non-IID features." In: *Asian Conference on Computer Vision (ACCV)*. Springer. 2022, pp. 41–55.

[89] Yanli Li, Dong Yuan, Abubakar Sadiq Sani, and Wei Bao. "Enhancing federated learning robustness in adversarial environment through clustering non-IID features." In: *Computers & Security* 132.1 (2023), pp. 1–13.

[90] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. "Backdoor learning: A survey." In: *IEEE Transactions on Neural Networks and Learning Systems* 1.1 (2022), pp. 1–18.

[91] Bin Liang, Hongcheng Li, Miaoqiang Su, Xirong Li, Wenchang Shi, and Xiaofeng Wang. "Detecting adversarial image examples in deep neural networks with adaptive noise reduction." In: *IEEE Transactions on Dependable and Secure Computing* 18.1 (2018), pp. 72–85.

[92] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. "Ensemble distillation for robust model fusion in federated learning." In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2020, pp. 2351–2363.

[93] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. "Fine-pruning: Defending against backdooring attacks on deep neural networks." In: *International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Springer. 2018, pp. 273–294.

[94] Lingjuan Lyu, Jiangshan Yu, Karthik Nandakumar, Yitong Li, Xingjun Ma, Jiong Jin, Han Yu, and Kee Siong Ng. "Towards fair and privacy-preserving federated deep

models." In: *IEEE Transactions on Parallel and Distributed Systems* 31.11 (2020), pp. 2524–2541.

[95] Cunqiang Ma, Bingsong Ma, Jiacai Wang, Zihao Wang, Xuan Chen, Binxing Zhou, and Xinghui Li. "Geographical origin identification of Chinese white teas, and their differences in tastes, chemical compositions and antioxidant activities among three production regions." In: *Food Chemistry: X* 16.1 (2022), pp. 100504–100514.

[96] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks." In: *International Conference on Learning Representations (ICLR)*. IMLS. 2018, pp. 1–23.

[97] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In: *Artificial intelligence and statistics (AISTATS)*. PMLR. 2017, pp. 1273–1282.

[98] Linhao Meng, Yating Wei, Rusheng Pan, Shuyue Zhou, Jianwei Zhang, and Wei Chen. "VADAF: Visualization for abnormal client detection and analysis in federated learning." In: *ACM Transactions on Interactive Intelligent Systems* 11.4 (2021), pp. 1–23.

[99] El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, Lê-Nguyên Hoang, and Sébastien Rouault. "Genuinely distributed byzantine machine learning." In: *Distributed Computing* 35 (2022), pp. 305–331.

[100] Umberto Michieli and Mete Ozay. "Are all users treated fairly in federated learning systems?" In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2021, pp. 2318–2322.

[101] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. "Agnostic federated learning." In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 4615–4625.

[102] Jose G Moreno-Torres, Troy Raeder, Rocıo Alaiz-Rodrıguez, Nitesh V Chawla, and Francisco Herrera. "A unifying view on dataset shift in classification." In: *Pattern Recognition* 45.1 (2012), pp. 521–530.

[103] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. "A survey on security and privacy of federated learning." In: *Future Generation Computer Systems* 115 (2021), pp. 619–640.

[104] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, J Doug Tygar, and Kai Xia. "Exploiting machine learning to subvert your spam filter." In: *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*. USENIX Association. 2008, pp. 1–9.

[105] Dinh C Nguyen, Ming Ding, Quoc-Viet Pham, Pubudu N Pathirana, Long Bao Le, Aruna Seneviratne, Jun Li, Dusit Niyato, and H Vincent Poor. "Federated learning meets blockchain in edge computing: Opportunities and challenges." In: *IEEE Internet of Things Journal* 8.16 (2021), pp. 12806–12825.

[106] Hung T Nguyen, Vikash Sehwag, Seyyedali Hosseinalipour, Christopher G Brinton, Mung Chiang, and H Vincent Poor. "Fast-convergent federated learning." In: *IEEE Journal on Selected Areas in Communications* 39.1 (2020), pp. 201–218.

[107] Thien Duc Nguyen, Phillip Rieger, Markus Miettinen, and Ahmad-Reza Sadeghi. "Poisoning attacks on federated learning-based IoT intrusion detection system." In: *Workshop on Decentralized IoT Systems and Security (DISS)*. ISOC. 2020, pp. 1–7.

[108] Sharnil Pandya, Gautam Srivastava, Rutvij Jhaveri, M Rajasekhara Babu, Sweta Bhattacharya, Praveen Kumar Reddy Maddikunta, Spyridon Mastorakis, Md Jalil Piran, and Thippa Reddy Gadekallu. "Federated learning for smart cities: A comprehensive survey." In: *Sustainable Energy Technologies and Assessments* 55 (2023), pp. 1–13.

[109] Qi Pang, Yuanyuan Yuan, Shuai Wang, and Wenting Zheng. "ADI: Adversarial Dominating Inputs in Vertical Federated Learning Systems." In: *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2023, pp. 1875–1892.

[110] Jungwuk Park, Dong-Jun Han, Minseok Choi, and Jaekyun Moon. "Sageflow: Robust federated learning against both stragglers and adversaries." In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2021, pp. 840–851.

[111] Balázs Pejó and Gergely Biczók. "Quality inference in federated learning with secure aggregation." In: *IEEE Transactions on Big Data* 9.5 (2023), pp. 1430–1438.

[112] Mary Phuong and Christoph Lampert. "Towards understanding knowledge distillation." In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 5142–5151.

[113] Frantisek Pollak, Peter Markovic, Jan Vachal, and Roman Vavrek. "Analysis of e-consumer behavior during the COVID-19 pandemic." In: *Intelligent Processing Practices and Tools for E-Commerce Data, Information, and Knowledge* (2022), pp. 95–114.

[114] Abinash Pujahari and Dilip Singh Sisodia. "Item feature refinement using matrix factorization and boosted learning based user profile generation for content-based recommender systems." In: *Expert Systems with Applications* 206 (2022), pp. 1–10.

[115] Youyang Qu, Md Palash Uddin, Chenquan Gan, Yong Xiang, Longxiang Gao, and John Yearwood. "Blockchain-enabled federated learning: A survey." In: *ACM Computing Surveys* 55.4 (2022), pp. 1–35.

[116] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.

[117] Shashank Rajput, Hongyi Wang, Zachary Charles, and Dimitris Papailiopoulos. "DETOX: A redundancy-based framework for faster and more robust gradient aggregation." In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2019, pp. 1–11.

[118] Irshad Ahmad Reshi, Shabir Ahmad Dar, and Shaikh Sobiya Ansar. "An Empirical Study on the Factors Affecting Consumer Behavior in the Fast-Food Industry." In: *Journal of Accounting Research, Utility Finance and Digital Assets* 1.4 (2023), pp. 376–381.

[119] Andrew Ross and Finale Doshi-Velez. "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients." In: *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI. 2018, pp. 1–10.

[120] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. "Distributed federated learning for ultra-reliable low-latency vehicular communications." In: *IEEE Transactions on Communications* 68.2 (2019), pp. 1146–1159.

[121] EK Sannara, François Portet, Philippe Lalanda, and VEGA German. "A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison." In: *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE. 2021, pp. 1–10.

[122] Yunus Sarikaya and Ozgur Ercetin. "Motivating workers in federated learning: A stackelberg game perspective." In: *IEEE Networking Letters* 2.1 (2019), pp. 23–27.

[123] Felix Sattler, Klaus-Robert Müller, Thomas Wiegand, and Wojciech Samek. "On the byzantine robustness of clustered federated learning." In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 8861–8865.

[124] Lloyd S Shapley et al. *A value for n-person games*. Princeton University Press Princeton, 1953.

[125] Richa Sharma, GK Sharma, and Manisha Pattanaik. "A CatBoost Based Approach to Detect Label Flipping Poisoning Attack in Hardware Trojan Detection Systems." In: *Journal of Electronic Testing* 38.6 (2022), pp. 667–682.

[126] Zebang Shen, Juan Cervino, Hamed Hassani, and Alejandro Ribeiro. "An agnostic approach to federated learning with class imbalance." In: *International Conference on Learning Representations (ICLR)*. IMLS. 2021, pp. 1–13.

[127] Marwaan Simaan and Jose B Cruz Jr. "On the stackelberg strategy in nonzero-sum games." In: *Journal of Optimization Theory and Applications* 11.5 (1973), pp. 533–555.

[128] Tianshu Song, Yongxin Tong, and Shuyue Wei. "Profit allocation for federated learning." In: *IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 2577–2586.

[129] Zhendong Song, Hongguang Sun, Howard H Yang, Xijun Wang, Yan Zhang, and Tony QS Quek. "Reputation-based federated learning for secure wireless networks." In: *IEEE Internet of Things Journal* 9.2 (2021), pp. 1212–1226.

[130] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. "Human activity recognition using federated learning." In: *IEEE Conference on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data &*

Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). IEEE. 2018, pp. 1103–1111.

[131] Jingwei Sun, Ang Li, Louis DiValentin, Amin Hassanzadeh, Yiran Chen, and Hai Li. "Fl-wbc: Enhancing robustness against model poisoning attacks in federated learning from a client perspective." In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2021, pp. 12613–12624.

[132] MA Syakur, BK Khotimah, EMS Rochman, and Budi Dwi Satoto. "Integration k-means clustering method and elbow method for identification of the best customer profile cluster." In: *Materials Science and Engineering* 336.1 (2018), pp. 1–7.

[133] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2015, pp. 1–9.

[134] Rahim Taheri, Reza Javidan, Mohammad Shojafar, Zahra Pooranian, Ali Miri, and Mauro Conti. "On defending against label flipping attacks on malware detection systems." In: *Neural Computing and Applications* 32 (2020), pp. 14781–14800.

[135] Rahim Taheri, Mohammad Shojafar, Mamoun Alazab, and Rahim Tafazolli. "FED-IIoT: A robust federated malware detection architecture in industrial IoT." In: *IEEE Transactions on Industrial Informatics* 17.12 (2020), pp. 8442–8452.

[136] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. "Data poisoning attacks against federated learning systems." In: *European Symposium on Research in Computer Security (ESORICS)*. Springer. 2020, pp. 480–501.

[137] Richard Tomsett, Kevin Chan, and Supriyo Chakraborty. "Model poisoning attacks against distributed machine learning systems." In: *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications (AIMLMDOA)*. SPIE. 2019, pp. 481–489.

[138] F Tramèr, D Boneh, A Kurakin, I Goodfellow, N Papernot, and P McDaniel. "Ensemble adversarial training: Attacks and defenses." In: *International Conference on Learning Representations (ICLR)*. IMLS. 2018, pp. 1–20.

[139] Xuezhen Tu, Kun Zhu, Nguyen Cong Luong, Dusit Niyato, Yang Zhang, and Juan Li. "Incentive mechanisms for federated learning: From economic and game theoretic perspective." In: *IEEE Transactions on Cognitive Communications and Networking* 8.3 (2022), pp. 1566–1593.

[140] Tiffany Tuor, Shiqiang Wang, Bong Jun Ko, Changchang Liu, and Kin K Leung. "Overcoming noisy and irrelevant data in federated learning." In: *International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 5020–5027.

[141] Jacobus Hendricus Van Lint. *Introduction to coding theory*. Springer Science & Business Media, 1998.

[142] Anusha Vangala, Ashok Kumar Das, Ankush Mitra, Sajal K. Das, and Youngho Park. "Blockchain-enabled authenticated key agreement scheme for mobile vehicles-assisted precision agricultural IoT networks." In: *IEEE Transactions on Information Forensics and Security* 18 (2023), pp. 904–919.

[143] Wei Wan, Jianrong Lu, Shengshan Hu, Leo Yu Zhang, and Xiaobing Pei. "Shielding federated learning: A new attack approach and its defense." In: *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2021, pp. 1–7.

[144] Guan Wang, Charlie Xiaoqian Dang, and Ziye Zhou. "Measure contribution of participants in federated learning." In: *IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 2597–2604.

[145] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. "Optimizing federated learning on non-iid data with reinforcement learning." In: *IEEE Conference on Computer Communications (INFOCOM)*. IEEE. 2020, pp. 1698–1707.

[146] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. "Attack of the tails: Yes, you really can backdoor federated learning." In: *Advances in Neural Information Processing Systems (NeurIPS)*. MIT Press. 2020, pp. 16070–16084.

[147] Lixu Wang, Shichao Xu, Xiao Wang, and Qi Zhu. "Addressing class imbalance in federated learning." In: *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI. 2021, pp. 10165–10173.

[148] Zheng Wang, Xiaoliang Fan, Jianzhong Qi, Chenglu Wen, Cheng Wang, and Rongshan Yu. "Federated learning with fair averaging." In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufmann. 2021, pp. 1–9.

[149] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Yo-Seb Jeon, and H. Vincent Poor. "Covert model poisoning against federated learning: Algorithm design and optimization." In: *IEEE Transactions on Dependable and Secure Computing* (2023), pp. 1–14.

[150] Shuyue Wei, Yongxin Tong, Zimu Zhou, and Tianshu Song. *Federated learning*. Springer, 2020.

[151] Hongda Wu and Ping Wang. "Fast-convergent federated learning with adaptive weighting." In: *IEEE Transactions on Cognitive Communications and Networking* 7.4 (2021), pp. 1078–1088.

[152] Han Xiao, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." In: *arXiv preprint arXiv:1708.07747* (2017).

[153] Han Xiao, Huang Xiao, and Claudia Eckert. "Adversarial label flips attack on support vector machines." In: *European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2012, pp. 870–875.

[154] Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. "Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation." In: *Uncertainty in Artificial Intelligence (UAI)*. PMLR. 2020, pp. 261–270.

[155] Cong Xie, Sanmi Koyejo, and Indranil Gupta. "Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance." In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 6893–6901.

[156] Cong Xie, Sanmi Koyejo, and Indranil Gupta. "Zeno++: Robust fully asynchronous SGD." In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 10495–10503.

[157] Zhaoping Xiong, Ziqiang Cheng, Xinyuan Lin, Chi Xu, Xiaohong Liu, Dingyan Wang, Xiaomin Luo, Yong Zhang, Hualiang Jiang, Nan Qiao, et al. "Facing small and biased data dilemma in drug discovery with enhanced federated learning approaches." In: *Science China Life Sciences* 65.3 (2022), pp. 529–539.

[158] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. "Verifynet: Secure and verifiable federated learning." In: *IEEE Transactions on Information Forensics and Security* 15 (2019), pp. 911–926.

[159] Miao Yang, Ximin Wang, Hongbin Zhu, Haifeng Wang, and Hua Qian. "Federated learning with class imbalance reduction." In: *European Signal Processing Conference (EUSIPCO)*. IEEE. 2021, pp. 2174–2178.

[160] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. "Federated machine learning: Concept and applications." In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (2019), pp. 1–19.

[161] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. "Applied federated learning: Improving google keyboard query suggestions." In: *arXiv preprint arXiv:1812.02903* (2018).

[162] Xin Yao, Chaofeng Huang, and Lifeng Sun. "Two-stream federated learning: Reduce the communication costs." In: *IEEE Visual Communications and Image Processing (VCIP)*. IEEE. 2018, pp. 1–4.

[163] Abbas Yazdinejad, Ali Dehghantanha, Reza M Parizi, Mohammad Hammoudeh, Hadis Karimipour, and Gautam Srivastava. "Block hunter: Federated learning for cyber threat hunting in blockchain-based iiot networks." In: *IEEE Transactions on Industrial Informatics* 18.11 (2022), pp. 8356–8366.

[164] Dongdong Ye, Rong Yu, Miao Pan, and Zhu Han. "Federated learning in vehicular edge computing: A selective model aggregation approach." In: *IEEE Access* 8 (2020), pp. 23920–23935.

[165] Hao Ye, Le Liang, and Geoffrey Ye Li. "Decentralized federated learning with unreliable communications." In: *IEEE Journal of Selected Topics in Signal Processing* 16.3 (2022), pp. 487–500.

[166] Fahri Anıl Yerlikaya and Şerif Bahtiyar. "Data poisoning attacks against machine learning algorithms." In: *Expert Systems with Applications* 208 (2022), pp. 1–20.

[167] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. "Byzantine-robust distributed learning: Towards optimal statistical rates." In: *International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 5650–5659.

[168] Felix Yu, Ankit Singh Rawat, Aditya Menon, and Sanjiv Kumar. "Federated learning with only positive labels." In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 10946–10956.

[169] Han Yu, Zelei Liu, Yang Liu, Tianjian Chen, Mingshu Cong, Xi Weng, Dusit Niyato, and Qiang Yang. "A fairness-aware incentive scheme for federated learning." In: *AAAI/ACM Conference on AI, Ethics, and Society (AIES)*. AAAI. 2020, pp. 393–399.

[170] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. "Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment." In: *International Conference on World Wide Web (WWW)*. ACM. 2017, pp. 1171–1180.

[171] Qunsong Zeng, Yuqing Du, Kaibin Huang, and Kin K Leung. "Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing." In: *IEEE Transactions on Wireless Communications* 20.12 (2021), pp. 7947–7962.

[172] Rongfei Zeng, Shixun Zhang, Jiaqi Wang, and Xiaowen Chu. "Fmore: An incentive scheme of multi-dimensional auction for federated learning in mec." In: *IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2020, pp. 278–288.

[173] Yufeng Zhan, Jie Zhang, Zicong Hong, Leijie Wu, Peng Li, and Song Guo. "A survey of incentive mechanism design for federated learning." In: *IEEE Transactions on Emerging Topics in Computing* 10.2 (2021), pp. 1035–1044.

[174] Hongpo Zhang, Ning Cheng, Yang Zhang, and Zhanbo Li. "Label flipping attacks against Naive Bayes on spam filtering systems." In: *Applied Intelligence* 51 (2021), pp. 4503–4514.

[175] Jingwen Zhang, Yuezhou Wu, and Rong Pan. "Incentive mechanism for horizontal federated learning based on reputation and reverse auction." In: *International World Wide Web Conference (WWW)*. ACM. 2021, pp. 947–956.

[176] Kaiyue Zhang, Xuan Song, Chenhan Zhang, and Shui Yu. "Challenges and future directions of secure federated learning: A survey." In: *Frontiers of Computer Science* 16 (2022), pp. 1–8.

[177] Zhengming Zhang, Ashwinee Panda, Linyue Song, Yaoqing Yang, Michael Mahoney, Prateek Mittal, Ramchandran Kannan, and Joseph Gonzalez. "Neurotoxin: Durable backdoors in federated learning." In: *International Conference on Machine Learning (ICML)*. PMLR. 2022, pp. 26429–26446.

[178] Ying Zhao, Junjun Chen, Jiale Zhang, Di Wu, Jian Teng, and Shui Yu. "PDGAN: A novel poisoning defense method in federated learning using generative adversarial network." In: *International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP)*. Springer. 2020, pp. 595–609.

[179] Xingchen Zhou, Ming Xu, Yiming Wu, and Ning Zheng. "Deep model poisoning attack on federated learning." In: *Future Internet* 13.3 (2021), pp. 1–14.

[180] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. "Federated learning on non-IID data: A survey." In: *Neurocomputing* 465 (2021), pp. 371–390.

[181] Tianqing Zhu, Ping Xiong, Gang Li, and Wanlei Zhou. "Correlated differential privacy: Hiding information in non-IID data set." In: *IEEE Transactions on Information Forensics and Security* 10.2 (2014), pp. 229–242.

[182] Rui Zhuang, Scott A DeLoach, and Xinming Ou. "Towards a theory of moving target defense." In: *ACM Workshop on Moving Target Defense (MTD)*. ACM. 2014, pp. 31–40.